Detecting and Resolving Referential Ambiguity in Requirement Specifications

S. Göktuğ Köse[a], Gürkan Demir[b] and Burak İ. Yıldız[c]

Boğaziçi University, Turkey

Abstract

Ambiguity is a type of language where a sentence is not explicitly defined, making multiple meanings possible. Different interpretations may lead to a misunderstanding of the requirements and thus have a significant effect on the development process's progress. Referential ambiguity arises when several precedents precede a pronoun. In this paper, we have proposed a rule-based approach to detect and resolve referential ambiguity in the fields of requirements specifications. According to results of experiments, our algorithm works well on given dataset. In the future, a machine learning model can be used in order to enhance performance on this task.

Keywords

Requirements Engineering, Referential Ambiguity, Ambiguity Detection, Ambiguity Resolution

1. Introduction

Ambiguity is a form of language in which there is no clear description of a sentence, argument or resolution, making multiple meanings possible. Ambiguities can be seen in the any fields of the life, and one of the area that ambiguities play significant role is requirements specifications. Different interpretations can lead to a misunderstanding of the specifications and thus have a major impact on the progress of the development process.

There are various types of ambiguities such as semantic, syntactic, referential. In referential ambiguity, when an anaphor can take its reference from more than one element, each plays the role of the antecedent. In other words, when a pronoun is preceded by multiple antecedents, referential ambiguity occurs. For example, the pronoun **they** in the below sentence can be referred to both **trucks** and **roads**.

*"The **trucks** shall treat the **roads** before **they** freeze."*

While in the elicitation process of requirements, it is very crucial to detect and resolve referential ambiguities due to the fact that those misunderstandings may lead project to fail. So it is very essential to have common understanding on what each requirement means.

CmpE 58D - Assignment 1: Detecting and Resolving Referential Ambiguity

salih.kose@boun.edu.tr (S.G. Köse); gurkan.demir@boun.edu.tr (G. Demir); ikan.yildiz@boun.edu.tr (B.İ. Yıldız)

In this paper, we have presented a solution for detecting and resolving referential ambiguity in the fields of requirements specifications. Our algorithm for this task is consisted of 2 parts:

1. **Detection**: Detecting the sentences that contain ambiguity.
2. **Resolution**: Finding the most likely antecedent for the sentences that are found as unambiguous.

The dataset contains 200 sentences that are manually marked as ambiguous or unambiguous by five annotators with experience in Software Engineering and/or Computational Linguistics. A sentence, which has an ambiguous referential case recognized by at least one of the annotators, or a disagreement on the meaning of that ambiguity is marked as nocuous. 102 sentences are categorized as ambiguous.

2. Related Work

In the field of requirements engineering (RE), most of the documents are written in natural language (NL) and processed with various natural language processing (NLP) techniques. The reason behind the widespread usage of NL is that software projects generally consists of multiple stakeholders and formal languages that is used in requirements modelling are not understandable for most of the stakeholders[1]. Even though usage of NL helps to increase the contribution of stakeholders to the projects, individuals may interpret the meanings of these documents differently because of the high risk of ambiguity in NL[1].

To overcome the ambiguity in NL, several researches were conducted NLP. Ng and Cardie[2] proposed a noun-phrase resolver by designing a comprehensive feature extraction methodology to improve the performance of existing machine-learning techniques. Soon et al.[3] proposed a machine learning(ML) approach to resolve noun phrase coreference by using morphological parsing, POS tagging, named entity recognition and semantic classifier. Bengston and Roth[4] have developed a baseline for more complex models by using a well-designed set of features along with a simple pairwise classification model to resolve coreference.

In the field of RE, several researches were conducted for detecting and resolving ambiguities in requirement specifications. Osman and Zaharin[5] proposed an automated approach to detect ambiguous requirement specifications and they have accomplished to classify influential words by using Random Forest classifier with using a list of words that yields best information gain. Kiyavitskaya et al.[6] discussed about the possible causes of ambiguity and proposed a two-step approach to identify ambiguities in requirement specifications by implementing a tool that applies a set of ambiguity measures to a requirement specification to identify ambiguous sentences and implementing another tool that would show ambiguities. Oo et al.[7] categorized the ambiguity detection in software requirement specification approaches into three groups as manual, semi-automatic using NLP techniques and semi-automatic using ML techniques by conducting an analysis on these techniques. As a result, they have pointed out the strengths and weaknesses of each technique and decided that Naïve Bayes classifier is the best fitting technique compared to both manual and NLP techniques in terms of accuracy and detection of ambiguity.

3. Method

First of all, we begin with downloading training data set and training answers set from ReqEval task's website and choose Google Colab as development environment. After downloading and integrating training data set with Colab, we read the data which is .xlsx format using Pandas Data Frame. Since training data set and answers data set in separate files, we start with merging them into one Pandas Data Frame and label ambiguous sentences as 0 and unambiguous ones as 1. Next, we have extracted the indices of the words which may cause ambiguity in the sentences marked as '<referential>' html tag. After that, we have added a new column to data frame which holds information of indices of words that will be checked for ambiguity. As a result, we have ended up with a data frame which has columns named sent_id, sent(abbreviation for sentence), decision(ambiguous/unambiguous), category, pronoun(word that may cause ambiguity), and sent_id_in_category.

Afterwards, we have defined singular and plural pronouns as separate lists. Single pronoun list corresponds to "i", "you", "he", "she", "it", "me","him", "her", "my", "your", "his", "its", "mine", "yours" ,"hers", "myself", "yourself", "himself", "herself", "itself"; plural pronoun list corresponds to "we", "you", "they", "us", "them", "our", "your", "their", "ours", "yours", "theirs", "ourselves", "yourselves", "themselves".

As the main part of our algorithm, firstly, we have extracted internals of sentences using spaCy en_core_web_sm model. This model extracted information about sentence word by word such as word is plural or singular, word is noun, pronoun, verb etc. The model can be reached via this link. Then, as a basis of our algorithm, we scan the words of the sentence before the referential pronoun word. If our pronoun is a singular pronoun we check whether there is exactly one noun that can be swapped with referential pronoun. If such, we accept the sentence as unambiguous and label it as 1. The same procedure is also applied for plural referential pronouns. After building basis of algorithm we examine the data set in more detail and add some additional grammar rules to eliminate misclassified examples. These rules includes general usages of English such as we try to eliminate sentences like "It is clear that...". In that sentence referential pronoun "It" shown in the beginning of the sentence and since we look for the words before the referential words, our algorithm was having difficulty on classifying such sentences. Besides these, we have also add additional rules to handle noun phrases including "of" keyword and examples in which gender difference should be considered such as he/she pronouns. While checking for nouns before referential pronoun, we have also stored nouns that can be swapped with referential pronoun in order to use in resolution part of the task.

In conclusion, we have tested our predictions with golden_resolution provided in the ReqEval task website. After evaluation, we have obtained 0.71 Precision and 0.90 Recall scores for detection part of the task. For the resolution, we have gained 0.24 precision, 0.24 recall and 0.53 precision, 0.53 recall scores for perfect matching and partial matching respectively. Additional information about evaluation can be found in Evaluation section.

Table 1
Detection Scores

Precision	Recall
0.71	0.90

4. Evaluation

To evaluate our rule-based system's performance, we have used the evaluator that was provided by the authorities of NLP4RE. As given in the task description, we had to extract two separate result sets.

1. **Detection**: A result set that shows the binary classification results of our system that a sentence whether ambiguous or unambiguous.
2. **Resolution**: A result set that shows the resolution of pronouns that are present in unambiguous sentences.

To obtain such result sets, we have applied several rules that we have discussed in the method section to the initial data set. After getting the initial result set of detection, we have compared with the golden result set that was provided by the authorities of NLP4RE by using the evaluator. We have realized that our rule-based approach did not perform well since we achieved a precision score of only 30% and misjudged most of the unambiguous cases as ambiguous. Therefore, we decided to advance our rule set. After advancing our rule set, we have re-evaluated our detection results. We have managed to achieve the precision and recall scores in Table 1 with our new rule set.

We have decided that these results we obtained in detection task were promising. Thus, we have also expected to achieve high scores in the resolution task considering the fact that the performance of detection task reflects to high scores in resolution task along with the performance of NLP tools that we have used.

To evaluate our rule-based pronoun resolver's performance, we have extracted nouns from the requirements specifications using spaCy's POS Tagger and tried to evaluate these nouns with the evaluator. We came up with a problem with the evaluator. Since our rule-based system did not perform with 100% accuracy, our result set contains resolution of sentences that are not present in the golden result set. To overcome this problem,

1. We have extracted the sentence ids from the golden resolution dataset and filled their resolution values with empty strings.
2. We have matched the ids that are present in our system's output and replaced empty strings with the corresponding resolution values.

Finally, we have obtained the precision and recall scores of our system that are shown in Table 2.

Table 2
Resolution Scores

	Precision	Recall
Perfect Matching	0.24	0.24
Partial Matching	0.53	0.53

Table 3
Simulated Resolution Scores

	Precision	Recall
Perfect Matching	0.35	0.35
Partial Matching	0.79	0.79

Our system performed poorly on perfect matching since the expected resolution differs from the output of spaCy parser. This does not mean that our rule-based system failed to resolve unambiguous terms considering the precision and recall scores with partial matching.

After that, we have evaluated our results with reducing the number of sentences both golden and actual resolution sets. The main idea behind this reduction is to create an intuition by simulating a hypothetical system that was achieved 100% accuracy on detection task.

The results in Table 3 indicate that perfect matching of resolution values still yields to low precision and recall scores because of the parsing differences that we have mentioned before. Precision and Recall scores that we have obtained with partial matching indicate that our resolution values are not completely irrelevant to golden values despite minor differences.

5. Conclusion

In this paper, we propose a rule-based solution for ReqEval Task: Ambiguity Detection and Disambiguation in Requirements Specifications to correctly detect and resolve referential ambiguities in requirements specification sentences. In principle, our rule-based approach utilizes well-known grammar rules and some additional rules defined above sections extracted after examining referential ambiguity errors in requirements specification sentences. The experimental results on training dataset provided by REFSQ2020 Workshop indicate that our approach precisely detect 71% of ambiguous sentences in dataset. Besides this, the approach has 90% Recall score for ambiguity detection. In addition to ambiguity, for the resolution part of the task, even though we have achieved comparatively lower scores on Perfect Matching, our approach are able to obtain up to 79% for Partial Matching scores. All experiment results can be seen in Table 1, Table 2, and Table 3. As a future work, since our approach is rule-based, it is straightforward to add additional rules to improve results as working on it.

6. References

[1] De Bruijn F, Dekkers HL. Ambiguity in natural language software requirements: A case study. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2010;6182 LNCS:233-47

[2] Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL '02). Association for Computational Linguistics, USA, 104–111.

[3] Wee Meng Soon , Hwee Tou Ng , and Daniel Chung Yong Lim. 2001. A Machine Learning Approach to Coreference Resolution of Noun Phrases. Computational Linguistics 27:4, 521-544

[4] Bengtson, Eric Roth, Dan. (2008). EMNLP'08 Understanding the Value of Features for Coreference Resolution. 294-303. 10.3115/1613715.1613756.

[5] M. H. Osman and M. F. Zaharin, "Ambiguous Software Requirement Specification Detection: An Automated Approach," 2018 IEEE/ACM 5th International Workshop on Requirements Engineering and Testing (RET), Gothenburg, Sweden, 2018, pp. 33-40.

[6] Kiyavitskaya, N., Zeni, N., Mich, L. et al. Requirements for tools for ambiguity identification and measurement in natural language requirements specifications. Requirements Eng 13, 207–239 (2008). https://doi.org/10.1007/s00766-008-0063-7

[7] Oo, Khin Nordin, Azlin Ismail, Amelia Ritahani Sulaiman, Suriani. (2018). An Analysis of Ambiguity Detection Techniques for Software Requirements Specification (SRS). International Journal of Engineering Technology. 7. 501. 10.14419/ijet.v7i2.29.13808.