Detecting and Resolving Referential Ambiguity

Sumeyye Agac[a], Yasemin Akan[a] and Busra Tabak[a]

[a]*Bogazici University, Istanbul*

Abstract

A clear and well-documented LaTeX document is presented as an article formatted for publication by CEUR-WS in a conference proceedings. Based on the "ceurart" document class, this article presents and explains many of the common variations, as well as many of the formatting elements an author may use in the preparation of the documentation of their work.

1. Introduction

Ambiguity is the capability of being understanding or interpreting something in more than one sense [1]. This problem occurs generally in natural language documents [2]. Requirements are usually specified in natural language and therefore, ambiguity is one of the main problems encountered in this area [3]. More clearly, in [4], they state that if a sentence is given to ten people and if this sentence is interpreted differently then, it is probably ambiguous.

Referential ambiguity is one of the subcategories of ambiguity problem. There are two important two terms in this domain which are antecedent and anaphora. The first mention is called antecedent and the second and any subsequent references is called anaphora. Detecting and resolving referential ambiguities manually is a boring and time-consuming process [2].

In this study, an automatic sentence basis referential ambiguity detection and Pronoun resolution system is proposed for English. The detection step consists of finding whether a sentence is ambiguous or unambiguous. The second step resolves unambiguous sentences. This is also called as pronoun resolution. It is one of the most important part of resolving referential ambiguity. The main idea consists pair an expression to its referring entity. The rest of the report is organized as follows: In section 1, we present related works. Then, in Section 2, we present the methodology of used to implement this system. In Section 3, we present the results of the system and investigate the impact of some parameters. Finally, in Section 5, we draw a conclusion.

sumeyye.agac@boun.edu.tr (S. Agac); yasemin.akan@boun.edu.tr (Y. Akan); busra.tabak@boun.edu.tr (B. Tabak)

2. Related Work

In [5], Gleich et al. developed an automatic ambiguity tool to detect the uncertainty of requirement attributes written in English or German, as well as educating needs authors on possible sources of uncertainty. The tool uses natural language processing techniques such as: part-of-speech tagging and regular expressions and detects 39 cases of ambiguity with 95% precision and 86% recall scores.

In [6], R Khezri proposed an automated tool called AmbiGO for detecting and resolving syntactic ambiguities in NL text. The tool detects and resolves three types of syntactic ambiguities: analytical, coordination, and PP attachment ambiguities. The author proposed three syntactic patterns, each to detect one type of ambiguity. First, the POS of the given sentence is calculated, then compared with the suggested phrases. Second, AmbiGO enhances a semantic analysis for the ultimate uncertainty solution. It takes the static possible readings corresponding to each pattern. After that, it gets the frequency numbers for each possible reading of an indefinite candidate from Google for semantic analysis. The tool achieved high precision of 100% for coordination and attachment, and 67% for analytical, but low recall of 23%, 15%, 17% for coordination, attachment, and analytical types.

In [7], Frederik S. Baumer et al. make a methodological contribution to software-aided improvement of user-defined requirement definitions by dealing with ambiguity and deficiency. Therefore, they have presented the data-driven language indicators that allow to optimize the common text analysis line through a needs-oriented analysis.

In [8], more recently, Mohamed Osama et al. proposed an automated approach to ambiguity detection and resolution. The approach detects syntactic ambiguity, including three different types at the sentence level, using syntactic analysis with the help of the Stanford CoreNLP API: coordination, attachment and analytical ambiguities. Although their approach performs well in ambiguous scenarios, it achieves a lower performance in non-ambiguous situations (defining them as ambiguous) due to lack of semantics.

Unlike these studies, we preferred to classify with logistic regression, naive bayes and logit boost methods using a feature set consisting of 15 features for the ambiguity detection system and to use a scoring system according to the features for our ambiguity resolution system.

3. Methodology

In this section, the methodology used is explained. Python 3.7 is used to implement ReqEval Task which consist of two parts: detection and resolution.

3.1. Dataset

The dataset is annotated by five annotators with an expertise in Software Engineering and/or Computational Linguistics. It contains 200 sentences, out of which 102 are marked as ambiguous. It consists of 6 domains which are spacecraft, digital home, archive file manipulation, weather, digital library and railway. The entire dataset will be split into ⅔ training set and ⅓ test set. The dataset can be downloaded [1].

The file 'training_set.csv' contains a sentence id represents the domain with a unique number and the sentence text. Some examples from the dataset are given below.

- "library#02", "The library may want to accept important digital materials in non-standard formats in case we are able to migrate <referential>them</referential> to a more usable format in the future."
- "weather#06", "CS shall collect environmental data based on <referential>its</referential> configured schedule."

As seen on the examples, each sentence contains at least one pronoun that is tagged with <referential>.

The answers files contain the same sentence id (as in the training set) and the corresponding solution according to the manual annotations. The "detection_answers_file.csv" contains the labels(ambiguous or unambiguous) and the "disambiguation_answers_file.csv" contains anaphors that are unambiguous and their referents.

3.2. Evaluation Metrics

For both detection and disambiguation tasks, we calculated accuracy and F1-scores. We also used precision and recall metrics to measure how well the system could identify and match the resolutions in the answers file.

$$Accuracy = \frac{TP + TF}{TP + TF + FP + FN} \quad (1)$$

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \quad (4)$$

3.3. Preprocessing

Text preprocessing is an important step for natural language processing (NLP) tasks. Tokenization, lemmatization, POS Tagging are the famous preprocessing steps. We used The Natural Language Toolkit, or more commonly NLTK, to apply them.

Tokenization is about splitting strings of text into smaller pieces, or "tokens". It is crucial to understand and find the pattern for various NLP tasks. Lemmatization is the process of

[1] https://drive.google.com/drive/folders/1P3chdoALzRmEdinHNj4OrVmqTnZlQ-b_.

converting a word to its base form. It usually refers to the morphological analysis of words.

The last preprocessing step is the process of classifying words into their parts of speech and labeling them accordingly is known as part-of-speech tagging, POS-tagging. This step is actually one of the most important steps for the solution that we present to our first detection task. Thanks to POS tagging, pronouns, nouns, conjunctions etc. in the sentences can be easily find.

3.4. Detection

In this part, the aim is to detect whether a sentence is ambiguous or unambiguous. In order to do that, we use two data files named training_set.csv and detection_answers_file.csv. The first has two columns while the first column contains sentence id's such as "weather#16" (16th sentence of weather domain), the second contains the sentence related to this id such as "QEDS shall log <referential>their</referential> activities." with referential tags. For the detection_answers_file.csv file, it has two columns. The first column is identic with the first column of training_set.csv, and the second column is the two classes of ambiguity which are "AMBIGUOUS" and "UNAMBIGUOUS". Since there are two classes, we thought the problem as a binary problem and decided to create a machine-learning based model.

We started by constructing some rules to create the feature vector of each sentence. Then, it will be used as an input of the machine learning algorithm. Given tokens and tags, for a sentence, the rules used and feature vector values associated with them are presented in Table 1.

Therefore, we have a feature vector of size 15 as input data and binary classes as labels (1: ambiguous and 0:unambigous). After creating feature vectors, to select properly a machine learning algorithm we tested our system using three classifiers (with default parameters) available in sklearn library. Firstly, we used Naïve Bayes and Logistic Regression algorithms to classify our data which are commonly used in NLP processes [9, 10]. Then, we investigated LogitBoost algorithm based on [11] which claims that it can work better than decision trees, J48, Naive Bayes, SVM, and Logistic Regression.

3.5. Resolution

In this part, firstly, we a pronoun resolution system. There are two important new terms in this domain which are antecedent and anaphora. The first mention is called antecedent and the second and any subsequent references are called anaphora. To give an example, in the sentence "The man is hungry, so he eats dinner.", the blue word is antecedent, and the green word is anaphora.

In English, there are two types of pronouns which are personal and demonstrative pronouns. Based on the data that we are interested in, we use personal pronouns in this design. The

Rule 1	There is at least one noun that appears before a pronoun	1 (yes), 0 (no)
Rule 2	Number of conjunctions	integer number
Rule 3	There is more than one pronoun	1 (yes), 0 (no)
Rule 4	There are at least two singular nouns.	1 (yes), 0 (no)
Rule 5	Number of verb is greater than four	1 (yes), 0 (no)
Rule 6	Number of punctuation marks is greater than two. (Marks are ',', '.', ';', '!', '?', ':', '"' and """)	1 (yes), 0 (no)
Rule 7	Number of nouns.	integer number
Rule 8	Number of pronouns	integer number
Rule 9	Number of verbs	integer number
Rule 10	Number of punctuation marks	integer number
Rule 11	There is at least one relative clause and one pronoun together	1 (yes), 0 (no)
Rule 12	There are at least two capital letters	1 (yes), 0 (no)
Rule 13	There are at least two prepositions	1 (yes), 0 (no)
Rule 14	There are at least two plural nouns.	1 (yes), 0 (no)
Rule 15	There is at least one word that cause ambiguity. (They are 'more', 'some', 'any', 'other', 'most', 'another', 'this', 'that', 'many' and 'certain')	1 (yes), 0 (no)

Table 1
Ambiguity detection rules

subcategories of personal pronouns are presented in Table 2.

There are several features used for this task such as number_agreement, gender_agreement and parallelism [12, 13, 14]. We designed a rule-based method where each feature is considered as a rule. When a rule is satisfied for a word pair in a sentence, the score associated with this pair increases. For example, considering a sentence with three words: "A <referential>B</referential> C." Based on the data file provided, we know that we search for B. Therefore, we look at the pairs (A, B) and (C,B). For each pair, we based on feature rules we give them some scores. In the end, we select the pair which has the highest score as the probably one. If score(A, B) is greater than the score (C, B), therefore we say that B refers to A and not B.

In order to obtain some features to investigate scores, we used StanfordCoreNLP tool. Given a text, it generates annotation of text. Among all annotations, we are particularly interested in "corefs" part which is more useful for our task. It contains id, text, type, number, gender, animacy, startIndex, endIndex, headIndex, sentNum, position and isRepresentativeMention [2]. Using some of them, we created three feature rules. Given word pair of candidate antecedent i

[2]https://stanfordnlp.github.io/CoreNLP/annotators.html

Table 2
Anaphora categories

Category	Anaphora
Nominative	he, she, it, they
Reflexive	himself, herself, itself, themselves
Possesive	his, her, its, their
Objective	him, her, it, them

Table 3
Precision and recall results (%) obtained by adding features step-by-step

Classifier	LOGISTIC REGRESSION		NAIVE BAYES		LOGIT BOOST	
	Precision	Recall	Precision	Recall	Precision	Recall
Rule 1	54.86	54.14	54.86	54.14	54.86	54.14
Rule 2	53.05	52.91	53.05	52.91	53.10	52.86
Rule 3	63.34	58.78	62.28	55.58	56.22	56.01
Rule 4	57.12	56.74	62.35	57.19	57.21	56.72
Rule 5	53.03	52.93	61.33	58.04	54.60	54.49
Rule 6	53.03	52.93	61.33	58.04	54.60	54.49
Rule 7	58.23	57.45	61.68	58.85	63.09	63.09
Rule 8	57.45	56.67	63.34	58.78	61.46	60.93
Rule 9	56.34	55.96	64.91	60.33	67.85	67.75
Rule 10	65.06	64.46	66.11	63.54	69.93	69.36
Rule 11	66.49	66.02	64.99	62.78	70.60	70.12
Rule 12	62.40	62.21	66.80	64.32	70.60	70.12
Rule 13	65.52	65.29	60.88	59.75	69.03	68.58
Rule 14	65.52	65.29	66.44	56.32	73.93	73.88
Rule 15	66.36	66.05	66.44	56.32	74.61	74.59

and reference/anaphora j:

- number_agreement: Add 1 to the score of the pair if i and j are both agree in number. This is useful to distinguish singular and plural entities and matched with singular/plural pronouns accordingly.
- gender_agreement: Add 1 to the score of the pair if i and j are both agree with respect to gender. For example, masculine, feminine and neuter pronouns in English differ because masculine and feminine pronouns refer to animate entities and neuter pronouns are always used to refer to inanimate entities.
- word_distance: Add 1 to the score of the pair if i is the closest candidate to the reference among all candidates.

After applying these rules, each word pair obtain a score between 0 and 3. Then the pair with the highest score is selected as antecedent-anaphora pair for related sentence.

```
TRUE -> the shunting leader <---> the shunting leader <-
FALSE -> the link assurance signal <---> this case <-
TRUE -> each authority <---> each authority <-
FALSE -> many of the railway specific standards <---> the railway specific standards referenced <-
FALSE -> the mobile <---> the engine <-
FALSE -> the user <---> the driver to terminate established calls which he is authorised to terminate <-
TRUE -> the users <---> users <-
FALSE -> nefu icp <---> the ne <-
TRUE -> tasks <---> tasks in a rate group <-
TRUE -> messages <---> messages <-
TRUE -> icps <---> the icps <-
TRUE -> redundancy management <---> redundancy management <-
ACCURACY:  57.8125  - TOTAL:  64  - TRUE PREDICTED:  37
```

Figure 1: A part of generated results

4. Performance Evaluation

4.1. Detection

Table 3 shows precision and recall results for detection task. The results are given for three classifiers for which we used, Logistic Regression, Naive Bayes and Logit Boost. These results are obtained by adding the feature vector along with the rules in Table 1. For example, in the first step, the feature vector contains only Rule 1. In the second step Rule 1 and Rule 2, in the third step Rule 1, Rule 2 and Rule 3 etc. In the last line, we used all the rules we defined. As can be seen from the Table, we can say in general that as the number of rules increased, the performance increased. However, some rules reduce the performance in intermediate steps. For example, adding Rule 5 after Rule 4 has reduced performance for all three classifiers.

Some rules increase the performance more than others. For example, Rule 10 for Logistic Regression, Rule 14 for Naive Bayes, Rule 9 for Logit Boost, Rule 3 and Rule 7 for three of them. However, as a result, the highest performance for Logistic Regression and Logit Boost was achieved by using the 15 rule together. The best performance for Naive Bayes is the step to which Rule 12 is added.

In the case that we have created a feature vector using all rules, accuracy and F1-Score are approximately 66.15% and 65.95% for Logistic Regression and 56.92% and 48.71% for Naive Bayes respectively. The scores obtained with Logit Boost, where we achieved the best performance, are 74.61% and 74.60%.

4.2. Resolution

To measure the success of our rule-based ambiguity resolution system, we first compared every data we found corresponding to the referential with the correct references in the 1ref_disambiguation_answers_file.csv file. Since we encountered some situations such as the correct reference is ”driver” and we find ”the driver”, we accepted them as if we found them correct. As a result of our scoring system, we were able to correctly find 37 references of 64 unambiguous

sentences in total. We calculated our accuracy with the formula true_prediction * 100 / total and reached 57.81%.

5. Conclusion

Detection and resolving ambiguity is one of the most important tasks in the requirement analysis process In this study, we investigated the implementation of a system related to this topic. In the first phase of the study, we implemented a machine learning-based ambiguity detection algorithm. In the second phase, for the sentences that are we found as unambiguous and also for the sentences that are labelled unambiguous, we performed a rule-based resolution.

The results show us that defining a sentence as ambiguous or unambiguous due to the nature of the problem is a challenging task. Therefore, using logit boost, we obtained a precision of 74.61% and recall of 74.59% which can be considered acceptable performance in such a problem. For the second phase, using the results of the first phase caused a negative impact as all of them are not detected correctly. Therefore, in order to better understand the performance of phase 2, we used only the sentences that are tagged as unambiguous in the original dataset. Therefore, we obtained an accuracy of 57.81%.

This study can be improved using more features and also taking into consideration the domain-specific properties. In order to experiment two strategy, we used a machine learning-based technique in the first phase and a rule-based technique in the second phase. However, the two techniques can also be merged to obtain better results further.

References

[1] J. Doe, Recommended practice for software requirements specifications (ieee), IEEE, New York (2011).

[2] M. Aref, et al., Detecting and resolving ambiguity approach in requirement specification: Implementation, results and evaluation, International Journal of Intelligent Computing and Information Sciences 18 (2018) 27–36.

[3] F. Chantree, B. Nuseibeh, A. De Roeck, A. Willis, Identifying nocuous ambiguities in natural language requirements, in: 14th IEEE International Requirements Engineering Conference (RE'06), IEEE, 2006, pp. 59–68.

[4] A. M. Davis, Software requirements: objects, functions, and states, Prentice-Hall, Inc., 1993.

[5] B. Gleich, O. Creighton, L. Kof, Ambiguity detection: Towards a tool explaining ambiguity sources, in: International Working Conference on Requirements Engineering: Foundation for Software Quality, Springer, 2010, pp. 218–232.

[6] R. Khezri, Automated detection of syntactic ambiguity using shallow parsing and web data (2017).

[7] F. S. Bäumer, M. Geierhos, Flexible ambiguity resolution and incompleteness detection in requirements descriptions via an indicator-based configuration of text analysis pipelines, in: Proceedings of the 51st Hawaii International Conference on System Sciences, 2018.

[8] M. Osama, A. Zaki-Ismail, M. Abdelrazek, J. Grundy, 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), IEEE, 2020, pp. 651–661.

[9] M. T. Naing, A. Thida, Pronominal anaphora resolution algorithm in myanmar text, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) 3 (2014) 2795–2800.

[10] D. Le Thanh, Two machine learning approaches to coreference resolution (????).

[11] H. Yang, A. Willis, A. De Roeck, B. Nuseibeh, Automatic detection of nocuous coordination ambiguities in natural language requirements, in: Proceedings of the IEEE/ACM international conference on Automated software engineering, 2010, pp. 53–62.

[12] K. Webster, M. Recasens, V. Axelrod, J. Baldridge, Mind the gap: A balanced corpus of gendered ambiguous pronouns, Transactions of the Association for Computational Linguistics 6 (2018) 605–617.

[13] G. Şahin, Impact of features on performance differences for gendered pronouns in pronoun resolution systems (-).

[14] R. Cuevas, I. Paraboni, Using 'low-cost'learning features for pronoun resolution, in: Proceedings of the 22nd Pacific Asia Conference on Language, Information and Computation, 2008, pp. 377–383.