Requirements for Tools for Ambiguity Identification and Measurement in Natural Language Requirements Specifications

Nadzeya Kiyavitskaya[1], Nicola Zeni[1], Luisa Mich[2], Daniel M. Berry[3]

[1]Department of Information and Communication Technologies,
University of Trento, Trento, Italy

[2]Department of Computer and Management Sciences,
University of Trento, Trento, Italy

[3]Cheriton School of Computer Science,
University of Waterloo, Waterloo, Ontario, Canada

nadzeya@dit.unitn.it, {nicola.zeni,luisa.mich}@unitn.it, dberry@uwaterloo.ca

Abstract

This paper proposes a two-step approach to identifying ambiguities in natural language (NL) requirements specifications (RSs). In the first step, a tool would apply a set of ambiguity measures to a RS in order to identify potentially ambiguous sentences in the RS. In the second step, another tool would show what specifically is potentially ambiguous about each potentially ambiguous sentence. The final decision of ambiguity remains with the human users of the tools. The paper describes two requirements-identification case studies with one small NL RS using a prototype of the first tool based on an existing NL processing system and a manual simulation of the second tool.

1 Introduction

Ambiguity is an intrinsic phenomenon of natural language. It means the capability of being understood in two or more possible senses or ways. Identification of ambiguous words and phrases is a crucial aspect in text-processing applications and many other areas concerned with human communication. The main focus of the present work is the problem of ambiguity identification in natural language documents, in particular with natural language (NL) requirements specifications (RSs) for computer-based systems (CBSs).

The main goals for any tool for identifying and measuring ambiguities in NL RSs are: (1) to identify which sentences in a NL RS are ambiguous and, (2) for each ambiguous sentence, to help the user to understand why it is ambiguous, so that he can remove the ambiguity from the sentence, and thus improve the NL RS.

There have been several attempts and proposals to apply linguistic tools to the requirements engineering (RE) problem of identifying and eliminating ambiguity in RSs for CBSs [26, 20, 9, 18, 19]. Despite the hopes raised by the success of such tools [e.g., 12, 23, 13, 17] in other domains, e.g., in message understanding, as evidenced by the annual Message Understanding Competition [14], these RE attempts have not been complete.

This paper proposes a two-step approach to identifying ambiguities in NL RSs. In the first step, one tool, T1, would be used to apply a set of ambiguity measures to a RS in order to identify potentially ambiguous sentences in the RS. In the second step, another tool, T2, would show what specifically is potentially ambiguous about each sentence in the RS. Since the final decision of whether a sentence is ambiguous rests with the human users of the tools, any sentence that either tool tags as potentially ambiguous is really only *potentially* ambiguous.

This paper describes work to identify requirements for T1 and T2. Implementation is left to later. Indeed, we are doing upfront RE for the tools to avoid implementing wrong requirements. By publishing this paper, we are in ef-

fect, inviting others to join in on the RE effort and even to implement what is required.

In this work, T1 was prototyped by shell scripts invoking commands offered by a general purpose NL processing (NLP) system, and it calculates a set of ambiguity measures that can be applied to the sentences of any NL RS, and for that matter, of any NL document. T2 was simulated by having the human authors of this paper search for instances of a collection of ambiguities identified in the literature as appearing in NL RSs.

Therefore, Section 2 of this paper reviews the ambiguity problem and the main work concerning ambiguity identification both for general text and for RS text. Section 3 describes T1 and a case study of its application to one particular small NL RS. Section 4 describes T2 and a case study of its application to the same NL RS used in the work described in Section 3. Conclusions are drawn in Section 5.

2 Overview of Ambiguity and Related Work

Ambiguity is a pervasive phenomenon in human languages, and is fundamentally a property of linguistic expressions. There are two basic definitions of "ambiguity":

1. the capability of being understood in two or more possible senses or ways;

2. uncertainty [3].

Uncertainty means lack of sureness about something, often because of gaps in the writer's background knowledge, the reader's background knowledge, or both. The issue of uncertainty is not considered in this paper; thus, the paper uses the first definition of "ambiguity".

A word, phrase, sentence, or other message is called *ambiguous* if it can be interpreted in more than one way. It is difficult to find a word that has only one meaning, and an isolated sentence, separated from its context, is, more often than not, ambiguous. Ambiguity gives NL its flexibility and usability, and consequently, it cannot be entirely eliminated from any NL.

Ambiguity in a RS can cause numerous problems. A RS for a CBS often forms part of a contract between the CBS's customer and supplier. Consequently, a RS needs to be precise, accurate, consistent, and complete, because it describes what the supplier must build. A RS needs to be clear enough to allow independent confirmation that a CBS built according to the RS does indeed satisfy the RS.

Clearly, any tool that can identify potential ambiguity in NL text might help the writer of a NL RS to create a high quality RS by showing her the potential ambiguities in her RS. Generally, no ambiguity identification tool can be perfect; it will fail to find some ambiguities, i.e., it will not have total *recall*, and it will find what are not really ambiguities, i.e., it will not have total *precision*; therefore, a tool can at best show only potential ambiguities. Once shown a potential ambiguity, the user can determine if the potential ambiguity is real, and if so, she can rewrite the problematic text. Of course, not all ambiguities can be easily identified. Finding some of them requires deep linguistic analysis.

The traditional types of ambiguity include lexical, syntactic, semantic, and pragmatic ambiguity. To this list we add two additional types, software-engineering, and language-error ambiguity [1]. Each of most of these types has subtypes. For more details on these types of ambiguity, see the survey by Berry and Kamsties [1].

Much work has been done in the field of ambiguity, and a number of linguistic theories have been developed. Resolving ambiguities is a requirement for many NL understanding applications. Indeed, for many in the NL understanding area, disambiguation *is* NL understanding.

Some work has considered the application of ambiguity identification in RE to help improve the quality of NL requirements specifications. The tools developed thus far use lexical and syntactical analysis to identify ambiguities and to measure the degree of ambiguity in a NL RS. Some of these tools try to measure also vagueness, subjectivity, optionality, and weakness of the RS. One class of tools are those developed specifically for NL RS ambiguity identification and measurement (NLRSAI&M). These include QuARS [9], ARM [26], KANT [22], and Chantree's tool [5]. Another class of tools are those developed for general linguistics purposes, but are applied to NLRSAI&M. These include LOLITA [23, 13, 20].

Other work has considered approaches to help RS writers to write less ambiguously, e.g., with patterns based on a metamodel of RS statements [7] or with a restricted language [10].

It should be noted that almost every tool requires some restrictions on the input NL text, even if only in the vocabulary used. Moreover, almost every approach assumes the user's involvement in the use of the approach's tool; the tool asks the user for help or presents to the user choices that must be made.

3 T1 and its Design and Construction Based on LOLITA

In the late 1990s, Luisa Mich and Roberto Garigliano developed T1, a LOLITA-based tool for calculating lexical, syntactic, and semantic ambiguities of words and sentences [20, 18]. They constructed T1 as a module by using commands of the LOLITA NLP system, which is a general-purpose, domain-independent NLP system designed for production use [11]. All the morphological, grammatical, semantic, pragmatic, and other data used by LOLITA

are stored in a large semantic net that serves as LOLITA's knowledge base. When LOLITA is provided with a NL document as input, LOLITA analyzes the document morphologically, syntactically, and then semantically. The semantic analysis yields a graph that is added to the semantic net.

Among the information that LOLITA determines for each parse tree t of a sentence S is the penalty of t as the intended parse tree of S. The penalty of a parse tree t of s is LOLITA's statement of how much effort it spent building t. This penalty is an attempt to model the likelihood for t to be the parse tree intended by the author of S. That is, the higher the penalty of t, the less likely that t is the parse tree the author intended. In addition, LOLITA offers to the user the tp command that can rank the parse trees of a sentence S according to each tree's penalty and output each tree with its penalty attached to it.

In LOLITA, the names and meanings of the specific penalty values from highest to lowest are:[1]

4: a tree with a penalty value of greater than or equal to 1000 has major structural problems, such as a missing or a repeated part of speech, e.g., zero or two verbs as in the phrase He verbs nouns and nouns verbs.[2],

3: a tree with a penalty value less than 1000 but greater than 100 has a major feature clash, such as an apparent or real dative or infinitive use of inappropriate verbs, e.g., I sent the user data. or I lent my son my maid.,

2: a tree with a penalty value less than 100 but greater than 30 has a minor feature clash, such as a wrong concordance, e.g., That is so twentieth century.,

1: a tree with a penalty value less than or equal to 30 but greater than 0 has at most some uncommon but nevertheless correct constructs, e.g., a noun used as an apposition to another noun, which is less common than an adjective used as apposition to a noun, and

0: a tree with a penalty value less than or equal to 0 has no problems whatsoever.

Thus, it is desirable to find at least one parse tree for S with its penalty value being less than or equal to 30. The scale of penalty values is exponential. Therefore, we have called each penalty value by a number proportional to the logarithm of the lower bound of the range the value is in, namely, the item numbers of the descriptions of the ranges given just above. Also, we collapse the range called "0" into the range called "1".

[1]The penalty values are given in the format "*name*: *meaning*".

[2]From now on, any example text is given in a sansserif typeface in order to reserve quotation marks for surrounding a quotation, the meaning of an example, and a non-example word used as itself. Morevoer, when an example ends with punctuation, that punctuation is given in the sansserif typeface and should be distinguished from possibly following punctuation, given in the serif typeface, that belongs to the surrounding sentence.

3.1 Ambiguity Measures Computed by T1

Using various LOLITA commands, T1 can calculate several measures on the words and on the sentences of the input document:

1. lexical ambiguity of a word W:
   $\alpha(W) =$ the number of meanings of W in the semantic net,

2. syntactic ambiguity of a word W:
   $\beta(W) =$ the number of syntactic roles, i.e., parts of speech, of W in the semantic net. Observe that for each word W, $\beta(W) \leq \alpha(W)$.

3. lexical[3] ambiguity of a sentence S:
   $\gamma(S) = \sum_{i=1}^{\#(S)} \alpha(S_i)$, where $\#(S)$ is the number of words in S and S_i is the ith word of S,

4. syntactic ambiguity of a sentence S:
   $\delta(S) =$ the number of parse trees of S reported by LOLITA's parser,

5. penalty of a parse tree t of a sentence S:
   $\pi(t, S) =$ the name of LOLITA's penalty range of t as a parse tree of S,

6. minimum penalty of a sentence S:
   $\pi(S) = \mathrm{Min}_{i=1}^{\delta(S)} \pi(t_i, S)$, where t_i is the ith parse tree among the $\delta(S)$ parse trees of S,

7. penalty-weighted syntactic ambiguity of a sentence S:
   $\delta^*(S) = \delta(S) \times \pi(S)$

8. lexical ambiguity of a word w in a sentence S according to a parse tree t of S:
   $\alpha^{t,S}(w) =$ the number of meanings of w in the semantic net that have the syntactic role r, where r is the syntactic role of w in t, which is a parse tree of S, and

9. syntax-weighted lexical ambiguity of a word w in a sentence S according to the parse trees of S:
   $\alpha^S(w) = \frac{\sum_{i=1}^{\delta(S)} \alpha^{t_i,S}(w)}{\delta(S)}$, where t_i is the ith parse tree among the $\delta(S)$ parse trees of S.

More details on these and other measures can be found in references 15 and 20.

[3]The literature calls this function "semantic ambiguity" for two reasons: (1) some approximate semantic ambiguity with lexical ambiguity, and (2) the word "lexical" applies to individual words and is somewhat meaningless when applied to a whole sentence.

Penalty-Weighted Ambiguity $= \delta^*(S)$	Sentence Subjected to LOLITA-Based Tool $= S$	Number of Trees $= \delta(S)$	Mininum Penalty Range ($\pi(S)$)
40	1. Customers select at least one video for rental.	10	$\geq$1000 (4)
10	1. Customers select at least one video to rent.	10	$\leq$30 (1)
8	2. The maximal number of tapes that a customer can have outstanding on rental is 20.	2	$\geq$1000 (4)
6	2. The maximal number of tapes that a customer can have on rental is 20.	2	$>$100 and $<$1000 (3)
14	3. The customer's *account* number is entered to retrieve customer data and create an order.	14	$\leq$30 (1)
4	3. The account number of the customer is entered to retrieve customer data and create an order.	4	$\leq$30 (1)
4	4. Each customer gets an id *card* from ABC for identification purposes.	1	$\geq$1000 (4)
4	5. This id *card* has a bar code that can be *read* with the bar code reader.	6	$\leq$30 (1)
?	6. Bar code Ids for each tape are entered and video information from inventory is displayed.	2	?
1	7. The video inventory file is updated.	1	$\leq$30 (1)
8	8. When all tape Ids are entered, the system computes the total bill.	2	$\geq$1000 (4)
2	9. Money is collected and the amount is entered into the system.	2	$\leq$30 (1)
1	10. Change is computed and displayed.	1	$\leq$30 (1)
2	11. The rental transaction is created, printed and stored.	2	$\leq$30 (1)
1	12. The customer signs the rental *form*, *takes* the tape(s) and leaves.	1	$\leq$30 (1)
?	13. To return a tape, the video bar code ID is entered into the system.	2	?
8	14. The rental transaction is displayed and the tape is *marked* with the date of *return*.	2	$\geq$1000 (4)
48	15. If past-due amounts are owed they can be paid at this time; or the clerk can select an option which updates the rental with the return date and calculates past-due fees.	12	$\geq$1000 (4)
32	16. Any outstanding video rentals are displayed with the amount due on each tape and the total amount due.	8	$\geq$1000 (4)
72	17. Any past-due amount must be paid before new tapes can be rented.	18	$\geq$1000 (4)

Table 1. LOLITA-Generated Data for ABCVPS Lines and Variants

3.2 Case Study with T1

Mich [18] describes the application of T1 on the ABC Video Problem statement (ABCVPS), a simple RS for a video tape rental system for the ABC Video company [6]. The center column of Table 1 shows the sentences of the ABCVPS, each sentence numbered and given on a separate line. When two lines share a number, the first is in the original ABCVPS, and the second is a variation tried during the experiment. The whole of Table 1 shows some of the syntactic ambiguity measures calculated from the outputs of the various commands of LOLITA applied to each of the sentences of the ABCVPS.

Each row shows the data for one sentence. The second column shows the sentence. The first column gives the penalty-weighted ambiguity, the δ^*, of the sentence. This value, which is the product of the values in the third and fourth columns, is given in the first column to allow quick determination of which sentences are regarded as most ambiguous. The third column gives the number of parse trees, the δ, for the sentence and the fourth column gives the range of the lowest penalty calculated for these parse trees, followed by the numerical name of the range, i.e., π of the sentence.

A word in a sentence S is italicized if the word has the highest syntax-weighted lexical ambiguity among all the words in all the parse trees of S. When a slight variation of an original sentence in the ABCVPS is given in the row underneath that of the original sentence, the variation is one that has fewer parse trees or a lower minimum penalty among its parse trees. The text that was replaced in the variation was regarded by LOLITA as making the sentence particularly ambiguous. In any row in which a "?" is given as the minimum penalty value, the tp command timed out when processing the row's sentence.

Notice the range of values in the first column, which shows the penalty-weighted ambiguity, δ^*, of each sentence. Each δ^* value ranges from a low of 1 to high of 72. From our experience, it seems right to classify a value of less than or equal to 5 as meaning "little or no ambiguity", a value of greater than 5 and less than 20 as meaning "somewhat ambiguous", and a value of greater than or equal to 20 as meaning "highly ambiguous".

There are a number of specific observations about the data in this table.

- Each of about half of the sentences has only 1 or 2 parse trees, well within what is considered little or no ambiguity.
- The data for each row was obtained by analyzing the row's sentence in the context of the complete list of sentences of the ABCVPS. In the case of a row that is a variation of one of the first three sentences, the context contains the original variation for the other of the first three sentences. The results would be worse if each sentence were analyzed separately, because then, no context information would be available.

Data obtained from many uses of LOLITA in many domains [13, 23] show that in general, each of about 20% of the sentences has only one parse tree. Thus, the ABCVPS is less ambiguous than the typical NL document.

A user of T1 may look at a sentence rated as highly ambiguous by T1 and ask what is ambiguous about it. This question should be answered by the proposed T2.

4 Requirements for T2

At the very least, T2 could exhibit for any sentence, all of its parse trees and all the word senses for each of its words. However, this output is not enough. There are serious problems with the ABCVPS that are not exposed by the current T1. The purpose of this section is to identify other ambiguity problems that should be exhibited by T2 when presented with a RS.

4.1 ABCVPS Case Study

To learn what these ambiguities are, we manually examined the ABCVPS to search for instances of a variety of problems mentioned in a variety of sources, including work by:

- Berry, Kamsties, and Krieger on ambiguities in NL RSs and legal contracts [3],
- Berry and Kamsties on the syntactically and semantically dangerous "all" and plural [3, 2],
- Bucchiarone, Fabbrini, Fusani, Gnesi, Lami, Pierini, and Trentanni on a model of the quality of RSs [9, 4],
- Denger on rules and patterns for high-quality RSs [7],
- Dupré on technical writing [8],
- Fuchs, Schwitter, and Schwertel on controlled English [10, 25],
- Kovitz on the style of RSs [16], and
- Rupp and Goetz on Neurolinguistic Processing [24].

Not all problems mentioned in these sources appear in the ABCVPS.

The list below gives the sentences of the ABCVPS. Each list item gives a sentence followed by an enumeration of the problems found in the sentence. Each problematic phrase[4] is bracketed and each pair of brackets has an index referring to an item in the enumeration of the problems in the sentence. Not all items in the enumeration of problems for a sentence are referred to by a bracket pair's index. Each

[4] "Phrase" is used in this section for "word or phrase" since a word is a degenerate phrase.

non-referenced item is a question involving more than one phrase or sentence. Only the first time that a particular problem occurs, a detailed explanation of the problem is given, surrounded by "≪" and "≫". In such an explanation, example text from the sentence whose problem is being explained is said to be from "the sentence at hand" so that examples from elsewhere can be addressed as "the example".

1. [Customers][(a)] select at least one video for rental.

   (a) Plural subject: ≪The problem with a plural subject is that in the absence of domain knowledge, it is not clear whether the complement of the verb applies to each instance of the subject or to plural subject as a whole [2, 25]. That is, in the sentence at hand, it is not clear whether each customer selects at least one video for rental or customers together select at least one video for rental.≫

2. The maximal number of tapes that a customer can have outstanding on rental is 20.

   (a) There is nothing in the ABCVPS that says that video and tape are synonyms. Domain knowledge tells us that they probably are synonyms. Moreover, the sentence at hand says that tapes can be outstanding on rental, while Sentence 16 of the ABCVPS talks about outstanding video rentals. ≪The problem with the presence of synonyms in a RS is that without domain knowledge, the reader cannot be confident that the synonyms mean the same. The problem is far more severe in an industrial strength RS written by several different people, each with his own set of synonyms for a concept. A solution is to decide on one term for each concept, that is, one representative from among each set of synonyms, and to use only that term or representative.≫

3. The customer's account number [is][(a)] entered to retrieve customer data and create an order.

   (a) Passive voice: ≪The problem with passive voice is that in the absence of domain knowledge, it is not clear who or what is doing the action [24]. This lack of clarity implies that it is not even clear whether (1) the environment does the action to the CBS or (2) the CBS does the action. In the former case, the requirement is for the CBS to *react to* the action. In the latter case, the requirement is for the CBS to *do* the action. This distinction is critical for writing the CBS's requirements correctly. Consequently, the sentence should be rewritten in active voice with an explicit subject doing the action.≫ The sentence at hand is truly ambiguous, because domain knowledge suggests that either the customer, an employee of ABC Video, or both could enter the customer's account number, e.g., by swiping the customer's id card in a bar code reader. Therefore, the requirements engineer would have to consult the customer about his or her desires in order to disambiguate the sentence at hand.

4. Each customer gets an id card from ABC for [identification purposes][(a)].

   (a) Weak phrase: ≪The problem with a weak phrase is that in the absence of domain knowledge, it is not clear what the phrase implies for the requirements of the CBS at hand [16].≫ In this case, what are the identification purposes? A solution is to replace the weak phrase with a more detailed phrase. The most likely meaning of for identification purposes in the sentence at hand is to identify the customer that is the subject of the sentence.

   (b) There is nothing that says that a customer gets only one id card from ABC. The sentence at hand says only that each customer gets an id card from ABC, and says nothing about making sure that a customer does not get more than one id card from ABC. Therefore, the ABC System needs to allow for a customer to have more than one id card.

5. This id card has a bar code that can be read with [the][(a)] bar code reader.

   (a) Noun with a definite article not introduced before: ≪The meaning of a noun preceded by a definite article, i.e., the, is that there is an instance of the denotation of the noun introduced in a previous sentence, by name or by use of an indefinite article, i.e., a, and that the instance with the definite article refers to that previously introduced instance [8].≫ The sentence at hand has the phrase the bar code reader. To what bar code reader is the phrase referring? None has been introduced in any previous sentence within the ABCVPS. Probably, the intent of the author of the sentence was to simultaneously introduce a bar code reader and to say that there is only one. The most direct way to achieve this intent is to say The ABC system has one bar code reader. From that sentence on, it is legitimate to talk about the bar code reader. If the uniqueness of the bar code reader is not required, then the author should say only The ABC system has a bar code reader. From that sentence

on, it is legitimate to talk about the bar code reader, but meaning only the one mentioned before.

(b) Nothing in the sentence at hand or even in the entire ABCVPS relates a customer's account number to the bar code of an id card that the customer has. Domain knowledge suggests that they are probably the same.

6. Bar code Ids for each tape [are][(a)] entered and video information from inventory [is][(b)] displayed.

   (a) Passive voice: Who or what enters bar code ids for each tape? A reasonable answer is an employee of ABC Video. However, with an automated system, the customer could very well enter bar code ids for each tape himself, by waving each tape in front of a bar code reader. Only the future owner of the ABC System can answer the question.

   (b) Passive voice: Who or what displays video information from inventory? The most likely answer is the ABC System.

7. [The][(a)] video inventory file [is][(b)] updated.

   (a) Noun with a definite article not introduced before: What video inventory file? If, as suggested in Item (c) below, video information from inventory and video inventory file are synonyms, then The video inventory file is the previously introduced video information from inventory

   (b) Passive voice: Who or what updates the video inventory file? The most likely answer is the ABC System, which is the subject of the ABCVPS RS.

   (c) Apparently, video information from inventory and video inventory file are synonyms.

8. When all tape Ids [are][(a)] entered, [the][(b)] system computes the total bill.

   (a) Passive voice: Who or what enters all tape Ids? A reasonable answer is an employee of ABC Video.

   (b) Noun with a definite article not introduced before: What system? The most likely answer is the ABC system that is the subject of the ABCVPS RS.

9. Money [is][(a)] collected and the amount [is][(b)] entered into the[(c)] system.

   (a) Passive voice: Who or what collects money?

   (b) Passive voice: Who or what enters the amount into the system?

   (c) This instance of a definite article is *not* bracketed because the system was introduced in the previous sentence.

   (d) What is the relationship between money and amount?

10. Change [is][(a)] computed and displayed.

    (a) Passive voice: Who or what computes and displays change? The most likely answer is the ABC System, which is the subject of the ABCVPS RS.

    (b) What is the relationship between change and what has appeared before?

11. [The][(a)] rental transaction [is][(b)] created, printed and stored.

    (a) Noun with a definite article not introduced before: What rental transaction? The most likely answer is that the rental transaction is the unique rental transaction being created, printed, and stored in the sentence at hand. In this case, *a* rental transaction is being created, printed, and stored.

    (b) Passive voice: Who or what creates, prints, and stores the rental transaction?

12. The customer signs [the][(a)] rental form, takes the tape(s) and leaves.

    (a) Noun with a definite article not introduced before: What rental form? The most likely answer is that the rental form is the rental transaction that is printed in the previous sentence and that rental form is a synonym for printed rental transaction.

    (b) Apparently, rental form and printed rental transaction are synonyms.

13. To return a tape, the video bar code ID [is][(a)] entered into the system.

    (a) Passive voice: Who or what enters the video bar code ID into the system?

    (b) Apparently, video and tape *are* synonyms because both words are used in the same sentence in a way that indicates that they are synonyms.

14. The rental transaction [is][(a)] displayed and the tape [is][(b)] marked with the date of return.

    (a) Passive voice: Who or what displays the rental transaction?

(b) Passive voice: Who or what marks the tape with the date of return?

(c) The physical tape is marked with the date of return? Domain knowledge suggests that the physical tape is not marked; rather the video information from inventory for the tape is changed to show the date of return.

15. If past-due amounts [are][(a)] owed they[(d)] can [be][(b)] paid at this time; or [the][(c)] clerk can select an option which updates the rental with the return date and calculates past-due fees.

(a) Passive voice: Who or what owes past-due amounts?

(b) Passive voice: Who or what can pay them, i.e., the past-due amounts, at this time?

(c) Noun with a definite article not introduced before: What clerk? There is no clerk mentioned before. Domain knowledge suggests that the clerk that was suddenly introduced in the sentence at hand is the mysterious employee of ABC Video that we had to invent to actively do the clerical functions of ABC Video that are expressed in passive voice.

(d) The they is *not* bracketed because it clearly refers to the immediately preceding plural noun phrase past-due amounts

(e) Should not the or following the semicolon be and?

(f) Are amounts and fees synonyms? After all, each can be past due. Domain knowledge suggests that indeed amounts and fees *are* synonyms.

(g) Both amount and amounts appear, the second being the plural of the first.

(h) Apparently clerk and employee of ABC Video are synonyms.

16. Any outstanding video rentals [are][(a)] displayed with the amount due on each tape and the total amount due.

(a) Passive voice: Who or what displays any outstanding video rentals with the amount due and the total amount due?

17. Any past-due amount must [be][(a)] paid before new tapes can [be][(b)] rented.

(a) Passive voice: Who or what must pay any past-due amounts before new tapes can be rented?

(b) Passive voice: Who or what can rent new tapes?

(c) What is the relationship between past-due amount and amount due on a tape and total amount due?

In summary, the most common problems were

1. the use of plural,
2. the presence of passive voice,
3. the presence of definite articles with no referents, and
4. the use of synonyms.

T2's detection of instances of Problems 1, 2, and 3 and problems similar to them requires that T2 has access to parse trees, parts of speech information, and other structural information about the sentences of T2's input RS. If also T2 were built based on LOLITA, this information would already be available from having run T1 on the same RS. Note that an instance of any of Problems 1, 2, and 3 could cause difficulties for the parser and other NLP modules and could result in multiple or incomplete parses for any sentence containing it.

Handling problem 4, requires discovery of synonyms. If T2 were based on LOLITA, then T2 would have access to LOLITA's semantic net, which uses also data previously gathered from the Web-accessible WordNet [21] thesaurus. To determine if a word w in a document is a synonym of another word in the same document, T2 could determine if any thesaurus synonym of w appears elsewhere in the document. Of course, the human user should be asked to confirm that any pair of automatically discovered synonyms is a pair of real synonyms.

Finally, the functionality of T1 has to be changed so that it shows these new kinds of ambiguity and uses some measure of the severity of each instance of these new kinds of ambiguity in computing the level of ambiguity of each sentence in a RS presented to T1.

4.2 **The** Only **Ambiguity**

The ABCVPS just happens not to have any example of the only ambiguity. However, its first sentence, The maximal number of tapes that a customer can have outstanding on rental is 20., could easily have been written using the word only, and most likely, the sentence would have been, A customer may only have 20 tapes outstanding on rental.. An informal survey of geographically close native-English-speaking colleagues of the authors confirmed that the given sentence is indeed the common only restatement of the original sentence. However, this only sentence is wrong, in that it does not say what the

sentence of which it is a translation says. The only sentence should be: A customer may have only 20 tapes outstanding on rental.. The mistaken only sentence says that the ony thing a customer may do with 20 tapes outstanding on rental is to have them, and certainly, the customer may not eat, smoke, burn, copy, or even *play* the 20 tapes outstanding on rental, unless it can be proved that these activities are part of the act of having.

The reason the sentence would most likely have been written A customer may only have 20 tapes outstanding on rental. is that the convention in English today is to put the only immediately preceding the main verb of the sentence, which is, in this case, have, regardless of where it should be put. The correct place to put the only is immediately preceding the word or phrase that is limited by the only, which is, in this case, 20 tapes. Interestingly, this convention of misplaced only seems to be only in English; in each other languages known to any of us, the word or phrase for only is placed before the word or phrase limited by the word or phrase for only. In English, words other than only suffer this misplacement problem. These other words include almost, also, even, hardly, just, merely, nearly, and really. Each of these words is a member of a class called *limiting words*. If the ABCVPS had any of these limiting words, the word would probably have been misplaced in any sentence containing it, and the sentence could have been the example of this subsection. The lack of one of these words in the ABCVPS notwithstanding, this misplaced word problem, particularly with the words only and also, occurs frequently in NL RS as well as in most technical papers[5].

5 Conclusions

This paper describes a two-step, tool-assisted approach to identifying ambiguities in NL RSs. In the first step, T1 would be used to apply a set of ambiguity measures to a RS in order to identify potentially ambiguous sentences in the RS. In the second step, T2 would show what specifically is potentially ambiguous about each sentence in the RS. The paper describes the use of a shell-script prototype for T1 and a manual simulation prototype for T2 for the purpose of exploring their requirements. Application of the prototypes to one small RS, the ABCVPS has shed some light on the requirements for T1 and T2. Additional experiments have been carried out to determine other requirements for T1 and T2 [15]. More work is needed to implement and evaluate the tools appropriately.

[5]This last sentence notwithstanding, this misplaced word problem does *not* occur in this paper, except in examples. The authors made sure of that!

Acknowledgments

Daniel Berry's work is sponsored in part by Canadian Grant NSERC-RGPIN227055-00 and in part by the Department of Computer and Management Sciences, University of Trento, Italy.

References

[1] D. M. Berry and E. Kamsties. Ambiguity in requirements specification. In J. Leite and J. Doorn, editors, *Perspectives on Requirements Engineering*, pp. 7–44. Kluwer, Boston, MA, USA, 2004.

[2] D. M. Berry and E. Kamsties. The syntactically dangerous *all* and plural in specifications. *IEEE Softw.*, 22(1):55–57, January/February 2005.

[3] D. M. Berry, E. Kamsties, and M. M. Krieger. From contract drafting to software specification: Linguistic sources of ambiguity. Technical report, Univ. of Waterloo, 2003. `http://se.uwaterloo.ca/~dberry/handbook/ambiguityHandbook.pdf`.

[4] A. Bucchiarone, S. Gnesi, and P. Pierini. Quality analysis of NL requirements: An industrial case study. In *Proc. of the 13th IEEE Int. Conf. on Reqs. Engr. (RE'05)*, pp. 390–394, 29 August–2 September 2005.

[5] F. Chantree. Ambiguity management in natural language generation. In *7th Ann. CLUK Research Colloquium*, January 2004.

[6] K. Cooper and M. Ito. SRRS training material. Technical Report CICSR-TR99-001, Univ. of British Columbia, 1999.

[7] C. Denger, D. M. Berry, and E. Kamsties. Higher quality requirements specifications through natural language patterns. In *Proc. of the IEEE Int. Conf. on Software-Science, Technology & Engr. (SwSTE'03)*, pp. 80–89. IEEE Comp. Soc. Pr., November 2003.

[8] L. Dupré. *Bugs in Writing: A Guide to Debugging Your Prose*. Addison-Wesley, Reading, MA, USA, second edition, 1998.

[9] F. Fabbrini, M. Fusani, S. Gnesi, and G. Lami. The linguistic approach to the natural language requirements, quality: Benefits of the use of an automatic tool. In *Proc. of the 26th Ann. IEEE Comp. Soc. - NASA GSFC Softw. Engr. Workshop*, pp. 97–105, 27–29 November 2001.

[10] N. E. Fuchs and R. Schwitter. Attempto Controlled English. In *CLAW'96, The 1st Int. Workshop on Controlled Language Applications*, 1996.

[11] R. Garigliano, B. Boguraev, and J. Tait. Editorial. *J. of Natural Language Engr.*, 1(1):1–7, 1995.

[12] R. Garigliano and D. J. Nettleton. Neo-pragmatism. In T. L. Group, editor, *The LOLITA Project: the First Ten Years, Vol. 3*. Springer Verlag, 1997.

[13] R. Garigliano, A. Urbanowicz, and D. J. Nettleton. Description of the LOLITA system as used in MUC-7. In *Proc. of the Message Understanding Conf. (MUC-7)*, 1998. `http://acl.ldc.upenn.edu/muc7/`.

[14] R. Grishman and B. Sundheim. Design of the MUC-6 evaluation. In *Proc. of the 6th Message Understanding Conf.*

(MUC-6), pp. 1–11, San Francisco, CA, USA, 1995. Morgan Kaufmann.

[15] N. Kiyavitskaya, N. Zeni, L. Mich, and D. M. Berry. Requirements for tools for ambiguity identification and measurement in natural language requirements specifications. Technical report, School of Computer Science, University of Waterloo, Waterloo, ON, Canada, 2007. `http://se.uwaterloo.ca/~dberry/FTP_SITE/tech.reports/KZMB2007AmbTR.pdf`.

[16] B. L. Kovitz. *Practical Software Requirements: A Manual of Content and Style*. Manning, Greenwich, CT, USA, 1998.

[17] L. Mich. NL-OOPS: From natural language to object oriented requirements using the natural language processing system LOLITA. *Journal of Natural Language Engineering*, 2(2):161–187, 1996.

[18] L. Mich. On the use of ambiguity measures in requirements analysis. In A. Moreno and R. van de Riet, editors, *Proc. of the 6th Int. Conf. on Applications of Natural Language to Information Systems (NLDB)*, pp. 143–152, 28–29 June 2001.

[19] L. Mich, M. Franch, and P. N. Inverardi. Requirements analysis using linguistic tools: Results of an on-line survey. *Reqs. Engr. J.*, 9(1):40–56, 2004.

[20] L. Mich and R. Garigliano. Ambiguity measures in requirement engineering. In Y. Feng, D. Notkin, and M. Gaudel, editors, *Proc. of Int. Conf. on Software—Theory and Practice (ICS2000), 16th IFIP World Computer Congress*, pp. 39–48, Beijing, 21–25 August 2000. Publishing House of Electronics Industry.

[21] G. A. Miller, C. Felbaum, and *et al*. *WordNet Web Site*. Princeton Univ., Princeton, NJ, USA, accessed 12 March 2006. `http://wordnet.princeton.edu/`.

[22] T. Mitamura. Controlled language for multilingual machine translation. In *Proc. of Machine Translation Summit VII*, 1999.

[23] R. Morgan, R. Garigliano, P. Callaghan, S. Poria, M. Smith, A. Urbanowicz, R. Collingham, M. Costantino, and C. Cooper. Description of the LOLITA system as used in MUC-6. In *Proc. of the 6th Message Understanding Conf. (MUC-6*, 1995.

[24] C. Rupp and R. Goetz. Linguistic methods of requirements-engineering (NLP). In *Proc. of the European Software Process Improvement Conf. (EuroSPI)*, November 2000. `http://www.iscn.com/publications/#eurospi2000`.

[25] U. Schwertel. Controlling plural ambiguities in Attempto Controlled English. In *Proc. of the 3rd Int. Workshop on Controlled Language Applications (CLAW)*, Seattle, WA, USA, 2000.

[26] W. M. Wilson, L. H. Rosenberg, and L. E. Hyatt. Automated analysis of requirement specifications. In *Proc. of the 19th Int. Conf. on Softw. Engr. (ICSE-97)*, pp. 161–171, New York, NY, USA, 17–23 May 1997. ACM Pr.