Ambiguity Detection: Towards a Tool Explaining Ambiguity Sources

Benedikt Gleich[1], Oliver Creighton[2], and Leonid Kof[3]

[1] Fakultät für Angewandte Informatik, Universität Augsburg,
Universitätsstr. 6a, D-86159 Augsburg, Germany
benedikt.nikolaus.gleich@student.uni-augsburg.de
[2] Siemens AG, Corporate Technology, Otto-Hahn-Ring 6, D-81730 München
oliver.creighton@siemens.com
[3] Fakultät für Informatik, Technische Universität München,
Boltzmannstr. 3, D-85748, Garching bei München, Germany
kof@informatik.tu-muenchen.de

Abstract. [**Context and motivation**] Natural language is the main representation means of industrial requirements documents, which implies that requirements documents are inherently ambiguous. There exist guidelines for ambiguity detection, such as the Ambiguity Handbook [1]. In order to detect ambiguities according to the existing guidelines, it is necessary to train analysts.
[**Question/problem**] Although ambiguity detection guidelines were extensively discussed in literature, ambiguity detection has not been automated yet. Automation of ambiguity detection is one of the goals of the presented paper. More precisely, the approach and tool presented in this paper have three goals: (1) to automate ambiguity detection, (2) to make plausible for the analyst that ambiguities detected by the tool represent genuine problems of the analyzed document, and (3) to educate the analyst by explaining the sources of the detected ambiguities.
[**Principal ideas/results**] The presented tool provides reliable ambiguity detection, in the sense that it detects four times as many genuine ambiguities as than an average human analyst. Furthermore, the tool offers high precision ambiguity detection and does not present too many false positives to the human analyst.
[**Contribution**] The presented tool is able both to detect the ambiguities and to explain ambiguity sources. Thus, besides pure ambiguity detection, it can be used to educate analysts, too. Furthermore, it provides a significant potential for considerable time and cost savings and at the same time quality improvements in the industrial requirements engineering.
Keywords: requirements analysis, ambiguity detection, natural language processing

1 Requirements Documents are Ambiguous

The overwhelming majority of requirements documents are written in natural language, as the survey by Mich et al. shows [2]. This implies that requirements are imprecise, as precision is difficult to achieve using mere natural language as the main presentation means. In software development, the later an error is found, the more expensive its correction is. Thus, the imprecision of requirements should be detected early in the development process.

Ambiguity (i.e., the possibility to interpret one phrase in several ways) is one of the problems occurring in natural language texts. An empirical study by Kamsties et al. [3] has shown that "... ambiguities are misinterpreted more often than other types of defects; ambiguities, if noticed, require immediate clarification". In order to systematize typical ambiguous phrases, Berry et al. introduced the Ambiguity Handbook [1]. A tool that detects ambiguities listed in the handbook surely contributes to early detection of problematic passages in requirements documents. According to Kiyavitskaya et al. [4], a tool for ambiguity detection should not only detect ambiguous sentences, but also explain, for every detected sentence, what is potentially ambiguous in it. Such a tool is presented in this paper.

Contribution: The tool described in the presented paper is a big step towards the ambiguity detection tool satisfying the requirements by Kiyavitskaya et al: this tool is able not only to detect ambiguities but also to explain ambiguity sources. When detecting ambiguities, it basically relies on a `grep`-like technique, which makes it highly reliable, applicable to different languages, and independent from error-prone natural language parsing. For every detected ambiguity the tool provides an explanation why the detection result represents a potential problem. Furthermore, due to web-based presentation and a lightweight linguistic engine on the server side, the tool is fast and highly portable, which makes it applicable for real projects. Therefore, it can cause considerable time and cost savings while at the same time enabling higher quality, as it simplifies early detection of potentially critical errors.

Outline: The remainder of the paper is organized as follows: First, Section 2 sketches part-of-speech tagging, the computational linguistics technique used in our tool. Then, Section 3 introduces the types of ambiguities that can be detected by our tool. These types include ambiguity classes introduced in the Ambiguity Handbook and ambiguity classes derived from writing rules used internally at Siemens. Section 4 presents the tool itself, especially the technique used to detect ambiguities and the presentation of the detected ambiguities to the tool user. Section 5 provides the results of the tool evaluation. Finally, Sections 6 and 7 present an overview of related work and the summary of the paper, respectively.

2 Computational Linguistics Technique: Part-of-Speech Tagging

Part-of-speech (POS) tagging marks every word of a given sentence with one of the predefined parts-of-speech (substantive, adjective, ...) by assigning a POS tag. For example, the words of the sentence "Failure of any other physical unit puts the program into degraded mode" are marked in the following way: `Failure°NN°failure of°IN°of any°DT°any other°JJ°other physical°JJ°physical unit°NN°unit puts°VBZ°put the°DT°the program°NN°program into°IN°into degraded°VBN°degrade mode°NN°mode`. Here, `NN` means a noun, `DT` a determiner, `JJ` an adjective, `VBZ` a verb, and `IN` a preposition. Additionally to part-of-speech tags, the tagger can provide the basic form for every word, as in the example presented above. Basic forms will be important for us to detect passive voice, where we will look for the verb "be". Following tags are the most important ones in the context of the presented work: (1) any tag starting with "`VB`", identifying different verb forms, (2) tag “`VBN`”, identifying verbs in the past participle form (“been”, “done”), (3) any tag starting with “`NN`”, identifying different noun forms, (4) tag `JJ`, identifying adjectives, and (5) tag `RB`, identifying adverbs. Complete information on tag meanings can be found in the official specifications of tagsets [5, 6]. Tagging technology is rather mature: there are taggers available with a precision of about 96%, such as the TreeTagger [7, 8], used in the presented work. The applied tagger (TreeTagger) provides support for English, German, and further languages, and thus allows to extend the presented work to really multilingual ambiguity detection.

3 Types of Ambiguities Detected by the Tool

The Ambiguity Handbook [1] lists several types of ambiguities, namely lexical, syntactic, semantic, and pragmatic ambiguities. Additionally, the Ambiguity Handbook states vagueness and language errors as further sources of problems. All these sources of problems are briefly introduced below. The citations in the below list are taken from the Ambiguity Handbook.

Lexical ambiguity: “Lexical ambiguity occurs when a word has several meanings.” This can be the case, for example, when one word has several meanings (like “green” meaning “of color green” or “immature”), or two words of different origin come to the same spelling and pronunciation (like “bank” meaning “river bank” and “bench”).

Syntactic ambiguity: “Syntactic ambiguity, also called structural ambiguity, occurs when a given sequence of words can be given more than one grammatical structure, and each has a different meaning.” This can be the case, for example, when the sentence allows different parse trees, like “small car factory” that can mean both “(small car) factory” and “small (car factory)”.

Semantic ambiguity: “Semantic ambiguity occurs when a sentence has more than one way of reading it within its context although it contains no lexical or structural ambiguity.” This can be the case, for example, when several quantifiers occur in the same sentence, like in “all citizens have a social security number” that can be interpreted in two ways:

- every citizen has an individual social security number:
  $\forall x.citizen(x) \Rightarrow \exists y.social\_security\_number(y) \wedge has(x, y)$
- all citizens have the same social security number:
  $\exists y.(social\_security\_number(y) \wedge \forall x.(citizen(x) \Rightarrow has(x, y)))$

Pragmatic ambiguity: “Pragmatic ambiguity occurs when a sentence has several meanings in the context in which it is uttered.” This can be the case when references occurring in the text can be resolved in several ways. For example, “they” in “The trucks shall treat the roads before they freeze” can refer both to the roads and to the trucks.

Vagueness: Vagueness occurs when a phrase has a single meaning from grammatical point of view, but still leaves room for interpretation, when considered as a requirement. “The system should react *as fast as possible*” provides an example of such a vague phrase.

Language error: Language errors represent grammatically wrong constructions, like "Every light has their switch." The above construction can be interpreted in different ways, both as "Every light has its own switch" and as "All lights have their common switch" and cause problems later.

Independently from the ambiguity type, we can apply ambiguity detection on the same four levels, namely lexical, syntactic, semantic, and pragmatic. These are the levels traditionally used in natural language processing, cf. [9]. Analysis tasks and result types for every kind of analysis are sketched in Table 1. A survey performed in our previous work [10] shows that solely lexical and syntactic analyses are possible at the moment for fully-fledged English. If the grammar used in the text can be restricted to a certain subset of English, semantical analysis becomes possible, too. Attempto Controlled English [11] gives an example of such a restricted language and a processing tool for this language. Pragmatic analysis is not possible yet. In order to keep our tool applicable to different documents, written without any grammatical restrictions, as well as to make the tool efficient, we focus on lexical and lightweight syntactical analysis, based on part-of-speech tagging. The applied analysis rules are presented below in Section 4.

Table 1. Classification of text analysis techniques

Analysis type	**Analysis tasks**	**Analysis results**
lexical	identify and validate the terms	set of terms used in the text
syntactic	identify and classify terms, build and validate a domain model	set of terms used in the text and a model of the system described in the text
semantic	build a semantic representation of every sentence	logical representation of every sentence, formulae
pragmatic	build a representation of the text, including links between sentences	logical representation of the whole text, formulae

The patterns for ambiguity detection have been extracted from the Ambiguity Handbook and an Siemens-internal guidelines for requirements writing. The Ambiguity Handbook lists a total of 39 types of ambiguity, vagueness or generality. Some of these patterns were not integrated into our tool: 4 out of 39 types were isolated cases, i.e., examples of ambiguous expressions without explicit statements, which linguistic patterns can be used to identify the ambiguity. "Mean water level" is an example of such an expression: to make it unambiguous, it is necessary to define precisely, how "mean" is determined, but we cannot generalize this expression to an ambiguity detection pattern. 7 further ambiguities are on a semantic or pragmatic level and are not amenable to state-of-the-art computational linguistics. Elliptic ambiguities like "Perot knows a richer man than Trump" provide an example of such a high level ambiguity. Lastly, ambiguities in formalisms (counted as one of 39 ambiguity types too) were not included in our tool, as we aim at the analysis of requirements written in natural language.

The remaining 27 patterns were integrated into our tool. In addition, all 20 patterns from the Siemens guidelines could be integrated, as they all can be easily detected on lexical or syntactic level. 9 out of 20 Siemens patterns are already covered by 8 patterns

Table 2. Ambiguity patterns with source and level of detection. Sources: AH=Ambiguity Handbook, S=Siemens

Ambiguity	Source	Ambiguity level	Level of detection
"up to", without explicit "including/excluding"	AH	semantic	syntactic
they	AH	pragmatic	lexical
everybody followed by their/its	AH, S	semantic	syntactic
Ambiguous words like include, minimum, or, …	AH	semantic, pragmatic	lexical
Vague words like acceptable, easy, efficient...	AH	semantic, pragmatic	lexical
Dangerous plural: all, each, every...	AH, S	semantic, pragmatic	syntactic
Dangerous plural with ambiguous reference (e.g. every ... a)	AH, S	semantic, pragmatic	syntactic
Both at the beginning of a sentence	AH	semantic, pragmatic	syntactic
many, few	AH	semantic, pragmatic	lexical
only, also, even	AH, S	syntactic	lexical
otherwise, else, if not	AH	semantic, pragmatic	lexical
not	AH, S	semantic, pragmatic	lexical
not because	AH	semantic, pragmatic	lexical
"and" and "or" in the same sentence	AH, S	syntactic, semantic	syntactic
until, during, through, after, at	AH	semantic, pragmatic	lexical
could, should, might	S	semantic, pragmatic	lexical
usually, normally	S	semantic, pragmatic	lexical
actually	S	semantic, pragmatic	lexical
100%, all errors	S	pragmatic	lexical
he, she, it	AH, S	pragmatic	lexical
brackets	S	syntactic, semantic, pragmatic	lexical
Slashes	S	syntactic, semantic, pragmatic	lexical
tbd, etc	S	semantic, pragmatic	lexical
fast	S	semantic, pragmatic	lexical
passive	S	semantic, pragmatic	syntactic, with part-of-speech tags
"this" (ambiguous reference)	AH, S	semantic, pragmatic	lexical
vague or ambiguous adjectives	AH	lexical, semantic and pragmatic	syntactic, with part-of-speech tags
vague or ambiguous adverbs	AH	lexical, semantic and pragmatic	syntactic, with part-of-speech taggs

from the Ambiguity Handbook, so we had the total of 27+20-9=38 patterns included in the tool. The detection patterns that were finally implemented in the tool are presented in Table 2. To make the table compact, we merged similar patterns to a single line of the table, so there is no 1:1 correspondence between table lines and patterns. Here, it is important to emphasize that many of the ambiguity detection patterns implemented in our tool represent semantic or pragmatic ambiguities, although we perform solely lexical and syntactic analysis. This is also easy to see in Table 2.

4 Ambiguity Detection and Presentation

Technically, our ambiguity detection tool is basically similar to the Unix tool `grep`: it investigates the input text line by line, checks whether the analyzed line matches certain regular expressions, and, in the case of matching, marks the found match (ambiguity) in the analyzed line. The tool implementation goes beyond pure `grep` by modularizing the definitions of ambiguity types and regular expressions used to detect these ambiguity types. This allows for tool extensions even by people not familiar with the tool architecture. Tables 3-5 present the keywords and regular expressions used for ambiguity detection. For this presentation, we use standard notation for regular expressions, with "." denoting any character, "|" denoting set union, and "*/+" denoting iteration.

The ambiguities are grouped by the detection techniques:

- Table 3 (lexical patterns) presents the ambiguities resulting from the fact that certain words always entail several interpretations. Such ambiguities can be detected on the lexical level, so Table 3 presents single keywords used for ambiguity detection. The only constraint when searching for these keywords is that we have to search for whole words only, and must ignore keywords occurring as constituents of longer words. Otherwise, the tool would match "*or*" with "m*or*e". Thus, the tool implements special measures for whole word matchings.
- Table 4 (word combinations) presents ambiguities resulting either from co-occurrence of certain words or from occurrence of certain word in special positions. As for Table 3, the tool matches whole words or phrases only.
- Table 5 (syntactic patterns) presents three ambiguities whose detection requires part-of-speech tagging: passive voice and usage of adjectives and adverbs. As the output of the TreeTagger consists of triples (e.g. `is°VB°be`), the regular expressions are rather complex, since they have to cover complete triples to ensure a correct presentation of the error messages.

  Passive voice: To detect passive voice, we proceed as follows: the basic idea is to search for the verb "to be", followed by the past participle form. This search pattern can be expressed in the regular expression "be.*VBN", where the tag `VBN` denotes the participle form. Explicit search for different forms of the verb "to be" is unnecessary, as the applied POS-Tagger provides not only the tag, but also the basic form for every word. The problem is, however, that the above regular expression matches too much in compound sentences. For example, in the sentence "it *is* an interesting idea that can *be* further *explored*", it matches the whole text piece from "is" to "explored". In order to make the match more precise, we refined the detection rule to the following: passive is detected by occurrence of the verb “to be”, followed by the past participle, but no further verbs are allowed to occur between “be” and the participle. Such a word sequence can be matched by the regular expression presented in Table 5.

Adjectives and adverbs: The list of vague adjectives and adverbs in the Ambiguity Handbook is incomplete, as it contains “etc”. When analyzing manual evaluations (cf. Section 5) we came to the conclusion that there are a lot more adjectives and adverbs that are perceived as ambiguous, than listed in the Ambiguity Handbook. So, we decided to trade in some precision for recall and to mark every adjective and every adverb as a potential ambiguity. Marking of adjectives as a potential ambiguity source is also in line with the statement by Rupp [12] that any adjective in comparative form (“better”, “faster”) can result in misinterpretations.

Tables 3-5 clearly show that most ambiguities result from single ambiguous words (not from word combinations) and, thus, can be detected on the lexical level. To apply the tool to German documents, we use the same regular expressions, with the only difference that the keywords are translated and the regular expression for passive detection is altered to fit German grammar.

Table 3: Keywords used to detect ambiguities on the lexical level

Keyword	**Matched ambiguity**	**Example**
they	Potentially unclear reference	... they can login to the system.
and, or, include, after, before, next, previous, minimum, maximum	List of ambiguous words from the Ambiguity Handbook	Users can be admins or normal users.
acceptable, accurate, appropriate, easy, efficient, essential, immediately, minimum, maximum, periodically, sufficient, user-friendly	List of vague words from the Ambiguity Handbook	The system shall provide acceptable security.
all, each, every, everybody	Potentially dangerous plural	Every student has his/her student id.
many, few	Many and few are vague expressions	Few users will use the system.
only, also, even	The meaning depends highly on the position.	Only admins can delete users vs. Admins can only delete users.
not	Specifications should be positive.	Passwords shall not be shorter than 6 characters vs. Passwords must be at least 6 characters long.
otherwise, else, if not	These expressions can refer to too many cases	Otherwise, the system shall display an error message.
not because	The negative reasons are hardly relevant.	This is not because of security concerns.

until, during, through, after, at	These expressions do not specify the “outside” behaviour.	Maintenance shall be performed on sundays vs. only on sundays.
could, should, might	These expressions are not concise.	The system should avoid errors.
usually, normally	Unnecessary speculation	The system should not display errors normally.
actually	Requirements shall avoid possibilities	Actually, this requirement is important.
100 percent, all errors	Wishful thinking	The system must be 100 percent secure.
he, she, it	Potentially unclear reference.	The system uses encryption. It must be reusable.
(,)	Unclear brackets	The system shall use HTML (DOC) documents.
/	Unclear slashes	The System shall use HTML/DOC documents.
tbd, etc	These expressions denote that something is missing	The system shall support HTML, DOC etc.
fast	Vague non functional requirement	The system shall be fast.
this	Potentially unclear reference.	This is very important.

Table 4. Regular expressions used to detect ambiguities resulting from word combinations

Regular expression	Matched ambiguity	Example
up to (?!including| excluding)	Up to with unclear inclusion	The system shall support up to five concurrent users.
everybody .* their, everybody .* its	Either language error or ambiguous plural	Everybody uses their login id.
ˆboth	At the beginning of the sentence, both has a unclear reference	Both should be documented.
(all|each|every) .* (a|his|her|its|their|they)	Dangerous plural with ambiguous reference	Every student thinks she is a genius.
and .* or, or .* and	The combination of “and” and “or” leads to unclear associativity	The system shall read HTML and PDF or DOC files.

The tool marks every found ambiguity occurrence either red or orange or blue, depending on the severity of the found ambiguity. This marking idea is similar to errors and warnings produced by most compilers: An ambiguity is marked red, if it definitely represents a problem, and either orange or blue, if it, depending on the context, can be potentially unambiguous, too, cf. Table 3. The difference between blue and orange is based on our experiments with the tool: patterns that get blue markings are more likely to result in false positives. For every sentence that contains a detected ambiguity, the tool places a pictogram next to the sentence. If the user clicks on the pictogram, he/she

Table 5. Regular expressions involving Part-of-Speech tags

Regular expression	**Matched ambiguity**	**Example**
\b\w+?°V[^°]*°be (\W[^°]+?°(?!VB.)[^°]*°[^]+?)* \W\w+?°VBN°\w+	Authors should state requirements in active form, as passive conceals who is responsible for the action.	The system will be tested.
\w+?°JJ.?°[^]+	All adjectives.	The system shall be fast and configurable.
\w+?°RB.?°[^]+	All adverbs.	The system shall save data permanently.

will get an explanation for every marking in the sentence under analysis. Additionally, the tool user gets a short explanation if he/she places the mouse pointer over the marked text. Figure 1 shows an example of the presentation of found ambiguities to the user.

5 Evaluation

In order to evaluate the tool, we created a reference data set consisting of approximately 50 German and 50 English sentences. The sentences were not specially crafted for the tool evaluation, but taken at random from real requirements documents. Table 6 shows an excerpt from the reference data set.

Table 6. Reference data set, excerpt

1. The system should be easy to use.
2. The system shall have a world class industrial design.
3. The system shall be as light as possible.

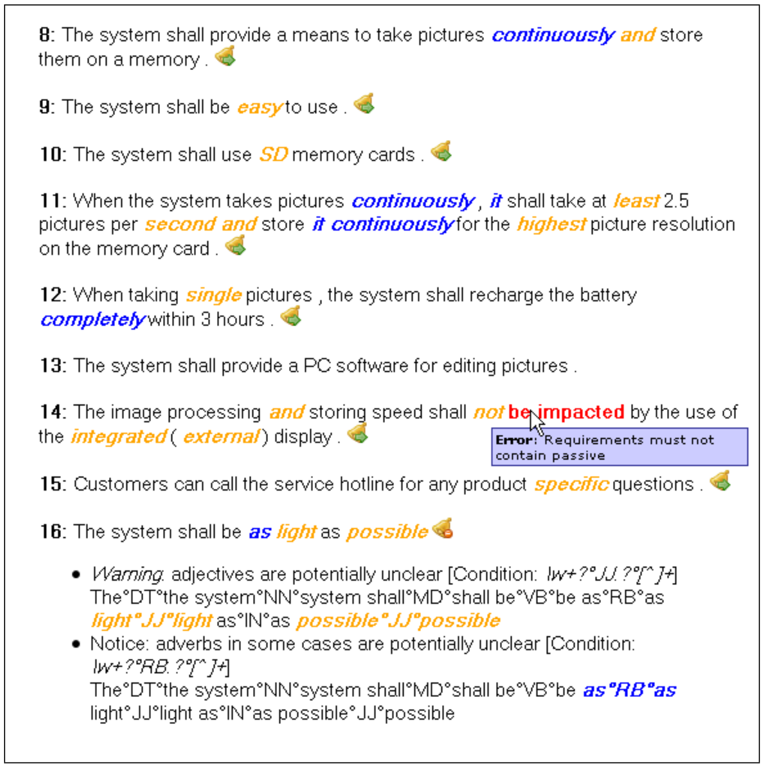
We asked 11 subjects to mark ambiguities in the data set. We had subjects from different backgrounds:

- 5 full-time requirements engineers and software consultants at Siemens,
- 2 software engineering master students at the University of Augsburg,
- 3 PhD students and 1 postdoc at the Technische Universität München, all doing research in requirements engineering.

The subjects were not trained in ambiguity detection, nor were they provided with theoretical backgrounds like the Ambiguity Handbook. Our subjects were given the following instructions[4]:

> You should find deficiencies in the provided text, especially ambiguities or imprecise passages, that negatively influence the requirements.
>
> The found errors can be marked on paper or submitted in some other way. It is important just to mark the text passage that you find problematic and to

[4] The original instructions were in German, here we provide an English translation



8: The system shall provide a means to take pictures *continuously* *and* store them on a memory .

9: The system shall be *easy* to use .

10: The system shall use *SD* memory cards .

11: When the system takes pictures *continuously* , *it* shall take at *least* 2.5 pictures per *second* *and* store *it* *continuously* for the *highest* picture resolution on the memory card .

12: When taking *single* pictures , the system shall recharge the battery *completely* within 3 hours .

13: The system shall provide a PC software for editing pictures .

14: The image processing *and* storing speed shall *not* **be impacted** by the use of the *integrated* (*external*) display .

Error: Requirements must not contain passive

15: Customers can call the service hotline for any product *specific* questions .

16: The system shall be *as* *light* as *possible*

- *Warning*: adjectives are potentially unclear [Condition: *\w+?°JJ.?°[^]+*]
  The°DT°the system°NN°system shall°MD°shall be°VB°be as°RB°as *light°JJ°light* as°IN°as *possible°JJ°possible*
- Notice: adverbs in some cases are potentially unclear [Condition: *\w+?°RB.?°[^]+*]
  The°DT°the system°NN°system shall°MD°shall be°VB°be *as°RB°as* light°JJ°light as°IN°as possible°JJ°possible

Fig. 1. Sample tool output, applied to the text used for evaluation (cf. Section 5)

estimate the criticality of the passage, on the scale from 1=very little relevance to 10=highly crucial.

The report on text deficiencies can have, the following form, for example:

. . . this should be actually done by the system . . .
Found problem: fuzzy phrasing; criticality: 3

Table 7 shows the same excerpt form the reference data set as Table 6, with markings by our subjects. It is easy to see that the same ambiguities were marked in different ways by different subjects: Subject 1 found the phrases from Table 6 unambiguous, Subject 2 marked single words or short phrases, and Subject 3 mostly marked extensive phrases. Furthermore, Subjects 2 and 3 attached different criticality values to the same ambiguities.

We considered an ambiguity as present in the evaluation text if it was marked by at least one subject, and defined the extents of the ambiguous phrase as the longest continuous word sequence marked by some subject. This means that we considered "easy to use" as the ambiguous phrase in Sentence 1 from Table 6. To define a criticality of a given ambiguity, we assigned the average criticality value provided by our subjects.

Table 7. Reference data set with markings by our subjects, excerpt

Subject	Markings	Criticality
Subject 1	1. The system should be easy to use.	—
	2. The system shall have a world class industrial design.	—
	3. The system shall be as light as possible.	—
Subject 2	1. The system should be easy to use.	7
	2. The system shall have a world class industrial design.	7
	3. The system shall be as light as possible.	7
Subject 3	1. The system should be easy to use.	10
	2. The system shall have a world class industrial design.	10
	3. The system shall be as light as possible.	8

The rationale for this decision was that we wanted to evaluate the reliability of the tool. More precisely, we wanted to evaluate whether manual analysis still remains necessary after tool application. In the case that the tool is absolutely reliable, the human analyst would solely have to decide whether potential ambiguities found by the tool are genuine ambiguities, but he/she would not have to search for other ambiguities manually. Thus, as a first approximation, it was necessary to treat *every* ambiguity marked by *some* subject as a genuine ambiguity.

To evaluate the tool performance, we used the following definitions: Let E be the set of ambiguities found by human evaluators, T be the set of ambiguities detected by the tool, and $S = T \cap E$. In the most simple form, we calculated recall and precision as $Precision = \frac{|S|}{|T|}$ and $Recall = \frac{|S|}{|E|}$. Additionally, we used the criticality values to calculate weighted recall. We defined $Recall_{weighted} = \frac{weight(S)}{weight(E)}$, where $weight(A) \overset{def}{=} \sum_{a \in A} criticality(a)$. Weighted precision makes no sense, as we would have to mix unrelated criticality values coming from different sources.

When calculating recall and precision according to the above definitions, we observed two interesting phenomena:

1. There exist text passages that were marked as problematic, but these markings represent no ambiguities but are purely stylistic. Furthermore, they are contradictory to explicitly stated best practices by Siemens or the suggestions of the Ambiguity Handbook. For example, one of our subjects marked “shall” as ambiguous, although the use of “shall” is not ambiguous at all and is even an explicitly stated best practice by Siemens. On the other hand, another subject marked every requirement that was in indicative mode, which is more a matter of taste than a real ambiguity. In the following definitions we will refer to such ambiguities as “$BP-$” (falsely marked ambiguities that are not ambiguities according to best practices). Here, it is important to emphasize that *no* ambiguity was absorbed into $BP-$ simply because it was not contained in the Ambiguity Handbook or Siemens guidelines. Marked ambiguities were absorbed into $BP-$ only if they were purely stylistic and in contradiction with best writing practices.

2. There exist ambiguities that were missed by every subject (thus, these ambiguities were not in E), which are still genuine ambiguities in the sense of the Ambiguity Handbook or the Siemens-guidelines. For example, many occurrences of passive voice were missed by the subjects. In the following definitions we will refer to such ambiguities as "$BP+$" (genuine ambiguities according to best practices, which were not found by our subjects).

In order to attenuate the influence of human evaluators' performance on the tool evaluation, we evaluated the tool not only with the original set of marked ambiguities (E), but also with $E \cup BP+$, $E \backslash BP-$ and $(E \cup BP+) \backslash BP-$ as reference sets. As the sets $BP+$ and $BP-$ are disjoint, it makes no sense to evaluate $(E \backslash BP-) \cup BP+$ separately, as it coincides with $(E \cup BP+) \backslash BP-$. Evaluation results are presented in Table 8.

Table 8. Evaluation results

Language	**Reference set**	**Precision (%)**	**Recall, simple (%)**	**Recall, weighted (%)**
English	E	47	55	64
	$E \cup BP+$	95	71	78
	$E \backslash BP-$	95	75	77
	$(E \cup BP+) \backslash BP-$	**95**	**86**	**86**
German	E	34	53	52
	$E \cup BP+$	97	76	74
	$E \backslash BP-$	97	69	70
	$(E \cup BP+) \backslash BP-$	**97**	**86**	**86**

For an ambiguity detection tool, the recall value is definitely more important than precision. In the ideal case, the recall should be 100%, as it would allow to relieve human analysts from the clerical part of document analysis [4]. For our tool, while the raw recall value of about 50% is rather low, the revised values ($(E \cup BP+) \backslash BP-$) of 86% show that the tool is fit for practical use, as it marks six of seven errors detected by humans and consistent with best practices. The precision is high, especially when $BP+$ is taken into account. Not every error detected by our tool is also detected by humans, but nearly every (95-97%) error found by the tool is based on a best practice from literature.

As for the errors marked by human evaluators but missed by the tool, manual analysis has shown that they represent either language errors, or pragmatic ambiguities where we could not identify explicit lexical or part-of-speech patterns that would allow to automate error detection. Language errors were presented by missing subject in two sentences: in "It assumed that for image call-up…" and in "Wants to have picture parameters…" A test with the C&C parser [13] has shown that one of these errors can be detected due to the incompleteness of the resulting parse (not yielding a complete parse tree), whereas the other would require a more thorough analysis of the resulting parse tree. Due to the computational complexity of the required analysis, C&C is rather slow, especially for interactive applications, when compared with the TreeTagger. Because of this and a rather small gain (at most two new errors from the reference data set would become detectable), we decided not to extend our tool in order to include parsing.

Pragmatic ambiguities not detected by the tool are more subtle: they occur in perfectly sensible and grammatically correct sentences, and profound knowledge of the application domain is necessary to spot the ambiguity. For example, our subjects marked ambiguities in following sentences:

- "If archiving of data failed even after a reasonable number of retries, the system shall store data locally and inform the user that it is currently not possible to archive data." (Evaluator's comment: what is the difference between archiving and storing?)
- "It assumed that for image call-up via a native client application installed on the Office PC there is the need to install some imaging components on the Office PC (primarily parts of the OEM application)." (Evaluator's comment: "need to install": who should perform the installation?)
- "The system shall allow taking at least 2 pictures per second." (Evaluator's comment: "When does a picture count as taken: when the button is pressed or when the picture is stored?")

Unfortunately, detection of such ambiguities is far beyond the capabilities of the state-of-the-art computational linguistics.

In addition to calculating the recall and precision of the tool, we calculated the recall of the human evaluators, taking E (set of ambiguities marked by at least one human evaluator) as the reference set. Surprisingly, it turned out that the average recall of a human analyst was at just 19%. Many of our subjects admitted, when submitting evaluation results, that attentiveness rapidly decreases when reading as little as 3 pages and that their markings were most probably influenced by this effect. However, a χ^2 test did not support the hypothesis that evaluators' performance decreased during reading of the provided text. Nevertheless, due the *perceived* performance decrease, an application of automated ambiguity detection can be helpful. Furthermore, although our tool is not perfect (recall below 100%), it detects a lot more genuine ambiguities than an average human analyst. Due to the large number of ambiguities that were not perceived by human analysts as such, it would be an interesting direction for future research, to investigate if a completely disambiguated text would still be perceived as a natural text.

6 Related Work

Lightweight text processing techniques (techniques not involving natural language parsing) are very popular in requirements analysis, as they are easy to implement and, nevertheless, can provide valuable information about document content. Such techniques can be used, for example, to identify application specific concepts. Approaches by Goldin and Berry [14], Maarek and Berry [15], and Sawyer et al. [16] provide good examples of concept extraction techniques: they analyze occurrences of different terms, and basing on occurrence frequency, extract application-specific terms from requirements documents. The focuses of these approaches and our approach are different, though: we do not perform any concept extraction but focus exclusively on ambiguity detection.

Ambiguity detection approaches are closer to the presented tool and should be analyzed more thoroughly. Apart from the approaches by Berry et al. [1] and Kiyavitskaya et al. [4], used as the basis for the presented tool, ambiguity detection approaches were introduced by Fabbrini et al. [17], Kamsties et al. [18], and Chantree et al. [19]. The approach by Fabbrini et al. introduces a list of weak words and evaluates requirements documents on the basis of weak word presence. Weak word detection is already included in our tool, and adding further weak words to the detection engine is just a matter of extending the weak words database. The ambiguity types classified by Kamsties et al. became a part of the Ambiguity Handbook later, so our tool already covers most ambiguities presented there. The approach by Chantree et al. deals exclusively with the coordination ambiguity. Our tool, although not specially designed to detect coordination ambiguity, is able to detect coordination ambiguity, too, and, in addition to that a lot more other types of ambiguities.

The tool presented in this paper has one important advantage when compared to other existing ambiguity detection approaches: It can not only detect ambiguities, but also explain the rationale for the detected ambiguity. Thus, apart from pure ambiguity detection, the presented tool can be used to educate requirements analysts, too.

7 Summary

Ambiguous requirements introduce conflict potential to a software project, as different stakeholders can interpret them in different ways, and then argue, whose interpretation is the correct one. One way to avoid such problems is to detect ambiguities early in requirements analysis. The presented tool, although performing lexical and syntactic analysis only, is able to detect ambiguities on all levels, from lexical to pragmatic. Although not able yet to detect all ambiguities, as listed in the Ambiguity Handbook, the tool represents an important milestone in the development of a tool completely satisfying the requirements to ambiguity detection tools by Kiyavitskaya et al. [4]: The presented tool is able not only to detect ambiguities, but also to provide explanations for detected ambiguities. This makes the presented tool suitable not only for ambiguity detection, but also for education purposes. In the industrial requirements engineering, these benefits have the potential for considerable time and cost savings while enabling a higher quality of requirements at the same time. The modular and lightweight design of our tool facilitates integration and customization for many practical applications.

Acknowledgments

We want to thank the participants of our empirical evaluation and other people who helped to improve this paper: Bernhard Bauer, Naoufel Boulila, Andreas Budde, Roland Eckl, Dominik Grusemann, Christian Leuxner, Klaus Lochmann, Asa MacWilliams Daria Malaguti, Birgit Penzenstadler, and Carmen Seyfried.

References

1. Berry, D.M., Kamsties, E., Krieger, M.M.: From contract drafting to software specification: Linguistic sources of ambiguity (2003) http://se.uwaterloo.ca/~dberry/handbook/ambiguityHandbook.pdf, accessed 27.12.2009.

2. Mich, L., Franch, M., Novi Inverardi, P.: Market research on requirements analysis using linguistic tools. Requirements Engineering **9** (2004) 40–56
3. Kamsties, E., Knethen, A.V., Philipps, J., Schätz, B.: An empirical investigation of the defect detection capabilities of requirements specification languages. In: Proceedings of the Sixth CAiSE/IFIP8.1 International Workshop on Evaluation of Modelling Methods in Systems Analysis and Design (EMMSAD'01). (2001) 125–136
4. Kiyavitskaya, N., Zeni, N., Mich, L., Berry, D.M.: Requirements for tools for ambiguity identification and measurement in natural language requirements specifications. Requir. Eng. **13** (2008) 207–239
5. Santorini, B.: Part-of-speech tagging guidelines for the Penn Treebank Project. Technical report, Department of Computer and Information Science, University of Pennsylvania (1990) (3rd revision, 2nd printing).
6. Schiller, A., Teufel, S., Stöckert, C., Thielen, C.: Guidelines für das Tagging deutscher Textcorpora mit STTS. Technical report, Institut fur maschinelle Sprachverarbeitung, Stuttgart (1999)
7. Schmid, H.: Probabilistic part-of-speech tagging using decision trees. In: Proceedings of the International Conference on New Methods in Language Processing. (1994) 44–49
8. Schmid, H.: Improvements in part-of-speech tagging with an application to german. In: Proceedings of the ACL SIGDAT-Workshop. (1995) 47–50
9. Russell, S., Norvig, P.: Communicating, perceiving, and acting. In: Artificial Intelligence: A Modern Approach. Prentice-Hall, Englewood Cliffs, NJ (1995)
10. Kof, L.: On the identification of goals in stakeholders' dialogs. In: Innovations for Requirement Analysis. From Stakeholders' Needs to Formal Designs: 14th Monterey Workshop 2007, Monterey, CA, USA, September 10-13, 2007. Revised Selected Papers. Volume 5320 of LNCS., Berlin, Heidelberg, Springer-Verlag (2008) 161–181
11. Fuchs, N.E., Schwertel, U., Schwitter, R.: Attempto Controlled English (ACE) language manual, version 3.0. Technical Report 99.03, Department of Computer Science, University of Zurich (1999)
12. Rupp, C.: Requirements-Engineering und -Management. Professionelle, iterative Anforderungsanalyse für die Praxis. Second edn. Hanser–Verlag (2002) ISBN 3-446-21960-9.
13. Clark, S., Curran, J.R.: Parsing the WSJ using CCG and log-linear models. In: ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics, Morristown, NJ, USA, Association for Computational Linguistics (2004) 103
14. Goldin, L., Berry, D.M.: AbstFinder, a prototype natural language text abstraction finder for use in requirements elicitation. Automated Software Eng. **4** (1997) 375–412
15. Maarek, Y.S., Berry, D.M.: The use of lexical affinities in requirements extraction. In: Proceedings of the 5th international workshop on Software specification and design, ACM Press (1989) 196–202
16. Sawyer, P., Rayson, P., Cosh, K.: Shallow knowledge as an aid to deep understanding in early phase requirements engineering. IEEE Trans. Softw. Eng. **31** (2005) 969–981
17. Fabbrini, F., Fusani, M., Gnesi, S., Lami, G.: The linguistic approach to the natural language requirements quality: benefit of the use of an automatic tool. In: 26th Annual NASA Goddard Software Engineering Workshop, Greenbelt, Maryland, IEEE Computer Society (2001) 97–105
18. Kamsties, E., Berry, D.M., Paech, B.: Detecting ambiguities in requirements documents using inspections. In: Workshop on Inspections in Software Engineering, Paris, France (2001) 68 –80
19. Chantree, F., Nuseibeh, B., de Roeck, A., Willis, A.: Identifying nocuous ambiguities in natural language requirements. In: RE '06: Proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06), Washington, DC, USA, IEEE Computer Society (2006) 56–65