Identifying Nocuous Ambiguities in Natural Language Requirements

Francis Chantree Bashar Nuseibeh Anne de Roeck Alistair Willis
Department of Computing, The Open University,
Milton Keynes, U.K.
{F.J.Chantree, B.Nuseibeh, A.DeRoeck, A.G.Willis}@open.ac.uk

Abstract

We present a novel technique that automatically alerts authors of requirements to the presence of potentially dangerous ambiguities. We first establish the notion of nocuous ambiguities, which are those that are likely to lead to misunderstandings. We test our approach on coordination ambiguities, which occur when words such as and *and* or *are used. Our starting point is a dataset of ambiguous phrases from a requirements corpus and associated human judgements about their interpretation. We then use heuristics, based largely on word distribution information, to automatically replicate these judgements. The heuristics eliminate ambiguities which people interpret easily, leaving the nocuous ones to be analysed and rewritten by hand. We report on a series of experiments that evaluate our heuristics' performance against the human judgements. Many of our heuristics achieve high precision, and recall is greatly increased when they are used in combination.*

1 Introduction

Ambiguity — where something can be interpreted in more than one way — is endemic in natural language, and is therefore a considerable problem for requirements that are written in natural language [4] [14]. If stakeholders interpret a requirement in different ways, this can result in an incorrect implementation. Gause and Weinberg recognise the crucial position that ambiguity has in requirements engineering (RE) [11], and Berry et al. suggest that unintended ambiguity is the "Achilles' heel" of software requirements specifications [4]. Ambiguity can be more intractable than other defects, such as incompleteness, and research has shown it more frequently results in misunderstandings [16].

Our work is driven by the desire to locate ambiguities that will lead to misunderstandings, and so to incorrect implementations: "ambiguity is characteristic of poor quality requirements, and poor quality requirements are characteristic of challenged projects" [6]. Locating ambiguities during requirements analysis is a relatively cheap solution to the problem, as the cost of fixing errors at later stages of a system's development process can be orders of magnitude higher [5]. However, locating errors is nontrivial, and even requirements documents that have been checked many times can still contain defects [12] [13].

As well as ambiguities that lead to misunderstandings, there are others which have only one obvious interpretation. We refer to the former as *nocuous* and to the latter as *innocuous*. Nocuous ambiguities can be both those recognised as being present — *acknowledged* ambiguities — and those that go undetected — *unacknowledged* ambiguities. The latter are especially dangerous as they are carried over into future stages of the system development process.

This paper presents a technique that helps requirements developers identify nocuous ambiguities by automatically identifying innocuous ambiguities, and reporting only the remaining nocuous ones; the latter can then be disambiguated, and maybe rewritten, at a later stage. Our technique therefore offers a partial solution, which we believe is more appropriate than attempting a fuller understanding of ambiguities [28]. Natt och Dag et al. cite industrial experience that motivates the need for such automated support for requirements management [22], and Gervasi and Nuseibeh suggest that this is both feasible and useful [12] [13].

To test our approach, we look at ambiguity arising from *coordinations*, which are recognised as "a pernicious source of structural ambiguity in English" [25]. Coordination ambiguities are structural in that alternative readings result from the different ways that sequences of words can be grammatically structured. They represent only a small proportion of the ambiguity that occurs in texts, but they provide a useful demonstration of how to distinguish nocuous from innocuous ambiguities. For instance, the requirement:

> Patients' conditions are inspected and recorded automatically

contains a coordination ambiguity which is nocuous: automatically could just as easily apply to both inspected and recorded as to just recorded, and a misunderstanding might result. Our techniques are transferable to other ambiguities.

Previous work on ambiguity in the requirements engineering literature, e.g. [16] [4] [17] [26] [9], has tended to focus on ambiguities that the authors have experienced as being a problem for requirements engineers. This is understandable, but it relies too greatly on the experience and wisdom of the authors concerned. The potential dangers of some specific types of ambiguity are not rigorously explored, and these may cause more frequent and unexpected misinterpretations than any one author can anticipate. Furthermore, previous research in RE has not fully explored the differentiations between nocuous and innocuous ambiguity and between acknowledged and unacknowledged ambiguity. The contribution of this paper is to address these issues in requirements documents, using coordination ambiguities as test data, by developing and adapting techniques from natural language processing (NLP). The paper therefore bridges the two disciplines of RE and NLP.

Our approach is to gather a set of actual requirements containing ambiguous coordinations from a corpus of requirements documents. Then we determine how nocuous these ambiguities are by making surveys of human judgements on them. We then use heuristics to automatically predict these human judgements; we combine these heuristics with aim of maintaining good precision but increasing recall. We utilise a cross-validation technique, whereby all data is used for both training and testing, to avoid obtaining unrealistically optimistic results for our data.

This paper is structured as follows. In Section 2 we provide some background for our research, including a discussion of inspection techniques, which bear some similarity to our approach. In Section 3 we describe our treatment of ambiguity in requirements, and introduce the ambiguity we use as out test case. We describe how we create our dataset in Section 4. In Section 5 we present the heuristics we have developed to predict innocuous ambiguity, and in Section 6 we describe how we combine these. We discuss some possible threats to the validity of our approach in Section 7. Sections 8 and 9 conclude the paper and present proposals for future work. All the examples we use are from our corpus of requirements documents.

2 Background

Ambiguity can be *avoided* in requirements by using techniques such as controlled languages [10] [1], style guides [20] [29], and references such as lexicons. Alternatively, ambiguity is *detected* in requirements by using techniques such as inspections, fit criteria, and test cases [16]. However, Kamsties observes that fit criteria and test cases generally only reduce *vagueness* and *generality*, and that requirements authors can be reluctant to use controlled languages and style guides [16]. (Generality and vagueness are similar to ambiguity, but cause misunderstandings due to lack of specificity rather than to widely differing meanings.) Those who call for the use of lexicons of domain-specific terminology, e.g. [15], generally expect them to be of limited size. But to be useful for addressing ambiguity, they should contain phrases (and in our case coordinations) that are common in requirements, and would be enormous even for a narrowly defined domain. Constructing and using such documents would be a highly unwelcome task.

Inspection techniques are known to be effective and efficient at reducing errors in requirements documents [31]. Inspection techniques have often been used to detect inconsistency and incompleteness in requirements, e.g. [36] [23], but not so often to detect ambiguities. Some that do address ambiguity use sets of heuristics or checklists [27] [11] [9], but tend to be rather wordy and time-consuming techniques: we hope to offer requirements engineers a quicker solution. Kamsties et al. observed that most inspection technique approaches which do consider ambiguity merely ask the question "is the requirement ambiguous?" [17]. This is true even for well-developed scenario-based approaches [23] [31]. This question will bring ambiguity to the readers' attention, but may not make them aware of the extent to which misinterpretation might occur.

Kamsties et al. present a study that investigates ambiguity in RE documents more thoroughly using inspections [17]. They report that inspection techniques can be more successful than using more formal methods. However, as with the aforementioned researchers, they are solely concerned with detecting ambiguities, and not with determining whether or not they need to be addressed, so their work is not directly comparable to our own. Also, they do not deal explicitly with structural ambiguities like coordination ambiguity. Sawyer et al. cite coordinations as a clear cause of potential ambiguity in RE documents [30], but to our knowledge these have not until now been subjected to systematic analysis by RE researchers. (Other fields have recognised the threat: in the legal sector murder cases have hinged upon the interpretation of coordinations [32].)

3 The Ambiguity Problem

Here we discuss our approach to the problem of ambiguity in text, but first we introduce the type of ambiguity that we use as our test case.

Coordination ambiguity can occur whenever a coordinating conjunction is used. In this paper we consider the *central* coordinating conjunctions [24] and, or, and and/or. These are widespread, and give ambiguity that is consistent enough to be used as our test case [24]. In the requirement:

> Display categorized instructions and documentation,

it is unclear what is categorised. This is due to uncertainty about the order in which the meanings of the words are

processed. If the coordination of instructions and documentation takes place before the effect of the external modifier categorized is considered, then both the instructions and the documentation are categorized. We call this a *coordination-first* reading. If the effect of the external modifier is considered before the coordination takes place, then only the instructions are categorised — a *coordination-last* reading. The uncertainty about which reading is intended will affect what type of display is implemented as a result of this requirement. We only include coordinations of this type, with one modifier and two coordinated phrases, in our dataset.

3.1 Nocuous and Innocuous Ambiguity

Ambiguity is often not a problem for humans: we are capable of using our knowledge of the world, and the context supplied by the words surrounding an ambiguity, to disambiguate that ambiguity. For instance, in the requirement:

> It is describing the size of vector-based inputs and outputs,

a coordination-first reading in this context is by far the most likely, with vector-based applying to both inputs and outputs. This is largely because we guess that inputs and outputs have similar characteristics. We say that such ambiguities have a *single reading*, as they are generally only read in one way, and are therefore innocuous ambiguities. Innocuous ambiguities will probably not lead to misunderstandings, and can therefore be left in text. Coordination ambiguities that are generally read as either coordination-first or coordination-last are innocuous ambiguities.

We say that ambiguities which people read in different ways have *multiple readings*, and are nocuous ambiguities. The sentence given in the introduction is an example of the multiple reading situation: both coordination-first and coordination-last readings are likely. An important aspect of nocuousness which we must now determine is whether people *acknowledge* that multiple readings are possible.

3.2 Acknowledged and Unacknowledged Ambiguity

Nocuous ambiguity can take two forms in our analysis. We call an ambiguity that readers realise is present in the text an *acknowledged ambiguity*. For instance, the phrase:

> Communication and performance requirements,

might be recognised as being ambiguous. It is unclear whether the requirements are of both performance and communication types, or just the performance type. This could lead to a misunderstanding if, for instance, the phrase was used as a job briefing. But, if at least one stakeholder spots the ambiguity, they can discuss what the most likely meaning is or contact the author to elicit the intended meaning.

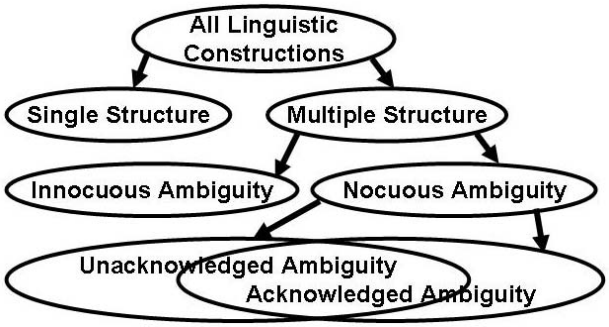

Figure 1. Multi-tier ambiguity representation

The ambiguous phrase can then be rewritten in a less ambiguous form, and incorrect implementation can be averted.

However, two or more readers can interpret a requirement differently, yet assume that their own interpretation is the only obvious one [3]. We call this phenomenon *unacknowledged ambiguity*. For instance, in the requirement:

> The user may define architectural components and connectors,

some might consider that architectural applies to both components and connectors, others that it only applies to components. If there are many types of components and connectors, this unacknowledged ambiguity may have consequences when the requirements are implemented. There is a risk that none of the stakeholders will acknowledge that this sentence contains a potentially misleading coordination ambiguity, and yet they have different readings of it. Then they will not clarify the meaning of the text, the text will not get re-written, and an incorrect implementation may occur.

Unacknowledged ambiguity is the same as *unrecognised disambiguation*: one of Gause's five most important sources of requirements failure [4]. The problem increases when stakeholders have different language abilities [4], e.g. at least one is not a native speaker of the language used. The RE community recognises the danger of unacknowledged ambiguity [17] [16], and that errors resulting from unconscious misunderstandings are surprisingly common [2].

Both acknowledged and unacknowledged ambiguities are nocuous because multiple readings are possible. It is important to capture acknowledged as well as unacknowledged ambiguities, as the former easily become the latter for any stakeholder who is prone to make minority interpretations. An ambiguity can be both acknowledged and unacknowledged in our approach, as these evaluations are made separately. The relationships between the ambiguity characterisations we have introduced is shown in Figure 1 — single and multiple structures are discussed in Section 4.1.

3.3 Notification not Disambiguation

Resolving ambiguities has never met with complete success, as it is difficult to supply computers with all the knowl-

edge necessary. Our approach is therefore to notify users of dangerous ambiguities, leaving them to perform disambiguation themselves. This uses the proficiency of computers at finding the ambiguities automatically, and the skill of requirements engineers' at deciding how to deal with them.

4 Creating The Dataset

We obtain our data by locating sentences containing coordinations in a corpus of requirements documents, and then evaluating their ambiguous. Ambiguity is subjective — a product of the meanings that people assign to language [34] — so rather than using absolute criteria we use human *judges* to determine ambiguity. We use many judges, rather than relying upon one person's opinion: this is known to be very effective (albeit expensive) for tasks such as ours [4].

4.1 Obtaining Suitable Requirements

We have built a corpus of requirements documents, obtained from colleagues and from the public domain. These documents specify systems from disparate domains, such as mechanical engineering, human computer interaction and telecommunications. In this corpus we identify sentences — including titles, bullet points, etc — that contain coordinations. We eliminate sentences containing coordinations having only one syntactic interpretation, i.e. which exhibit *single structure* rather than *multiple structure*, as shown in Figure 1. This accounts for a large majority of the sentences. Common reasons that a coordination has single structure are: it conjoins two sentences, no modifier is present, or the second coordinated phrase begins with an *article*. A full explanation of the criteria we use for identifying single structure coordinations is found in [8]. We have developed specialised *chunking* software to assist with the task of identifying the coordinations that interest us. This does not yet give highly precise results, but can be used to reliably recall all the coordinations we require along with others that must then be weeded out by hand.

For this study, we located 639 sentences containing multiple structure coordination ambiguity with and, or or and/or in our RE corpus — just over a quarter of the sentences there. We include sentences in our surveys regardless of how obviously preferable one particular reading may be: such a lack of discrimination is necessary in order to create realistic data for training and testing. Sentences containing two or more coordination ambiguities are split into separate sentences. After this process, we had 138 sentences for this study, each containing a single multiple structure coordination ambiguity. Each sentence can be considered to represent a requirement. The sentences we use in this study, and the raw corpus data from which they are derived, can be viewed at `http://mcs.open.ac.uk/fjc44/Research/Data.htm`

Table 1. Computing whether or not a sentence contains a nocuous ambiguity

For sentences $(1 \ldots i \ldots n)$

CF_i = No. of Coordination First Judgements on Sentence i

CL_i = No. of Coordination Last Judgements on Sentence i

QUA_i = Quotient Unacknowledged Ambiguity for Sentence i

$$= \frac{min(CF_i, CL_i)}{(CF_i + CL_i)}$$

A_i = No. of Ambiguous Judgements on Sentence i

AA_i: Sentence i is Judged an Acknowledged Ambiguity

IFF ($A_i \geq CF_i$) AND ($A_i \geq CL_i$)

UA_i: Sentence i is Judged an Unacknowledged Ambiguity

IFF $\left(QUA_i > \frac{\sum_{i=1}^{n} QUA_i}{n} \right)$

Sentence i is Judged a Nocuous Ambiguity IFF AA_i OR UA_i

4.2 Obtaining Judgements

The 138 requirements obtained from our corpus were shown to 17 judges in 4 separate surveys. Our judges are a group of computing professionals, comprising developers, academics and research students. They were asked to judge whether each coordination was to be read coordination-first, coordination-last, or if it was "ambiguous so that it might lead to misunderstanding". The 138 requirements, together with the judgements on them, form our dataset.

We now determine whether each requirement contains nocuous or innocuous ambiguity. We use a method that weights unacknowledged ambiguity as being more important than acknowledged ambiguity, as the former is more immediately dangerous. We say that a requirement contains an acknowledged coordination ambiguity if that ambiguity is judged to be ambiguous at least as often as it is judged to be coordination-first and at least as often as it is judged to be coordination-last. We say that a requirement contains an unacknowledged coordination ambiguity if it contains an above average percentage unacknowledged ambiguity. An ambiguity is nocuous if it is either an acknowledged ambiguity or an unacknowledged ambiguity, and it is innocuous if it is neither. This is detailed formally in Table 1.

This dividing line between nocuous and innocuous ambiguity is not the only one that can be used. For highly safety-critical applications one could classify as innocuous only those coordinations judged overwhelmingly to be either coordination-first or coordination-last. Or, with less critical applications, and when stakeholders have reliable language skills, one might allow more ambiguous coordinations to be considered innocuous. This flexibility of classification can be achieved using *ambiguity thresholds* [7].

5 Heuristics

Here we describe the individual heuristics we use to predict innocuous ambiguity. We then describe how we combine them, aiming for coverage which is the union of their individual coverages while maintaining good precision.

True positives in this study occur when our heuristics correctly predict that requirements contain innocuous coordination ambiguities. We use measures of precision and recall to evaluate how successful our heuristics are at achieving true positives and avoiding false results, and we use f-measure [33] to combine precision and recall.

$$Precision = \frac{No.\,of\,True\,Positives}{No.\,of\,Positive\,(Innocuous)\,Results\,by\,Heuristic}$$

$$Recall = \frac{No.\,of\,True\,Positives}{No.\,of\,Requirements\,Judged\,Innocuous\,in\,Surveys}$$

$$F\text{–}Measure = \frac{(1+\beta) * Precision * Recall}{\beta^2 * Precision + Recall}$$

Precision is much more important to us than recall: we wish our heuristics to indicate reliably how coordinations should be read. We therefore use a weighting of β = 0.25 for our f-measure, strongly in favour of precision; our aim is to maximise this score for the combined heuristics. For each heuristic we choose a *cut-off* point that maximises that heuristic's contribution to the performance of the combined heuristics. The cut-off, which is found experimentally for each heuristic, is a figure at or below (or above) which a heuristic's results are deemed to be positive.

We employ 10-fold cross validation, which ensures that statistics for a small set of data are not biased [35]. It avoids overfitting, whereby results are maximised unrealistically for a particular set of data. Our dataset is first randomly sorted, and then split into ten equal parts. Nine of the parts are concatenated and used for training to find the optimal cut-off for each heuristic. The combined heuristics are then run on the held-out tenth part using those cut-offs. This procedure is carried out iteratively with a different held-out part each time and different cut-offs proving to be optimal. The performances on all the held-out parts are then averaged to give figures for the combined heuristics.

Most of the heuristics we consider use the intuition that some form of similarity between the coordinated phrases will indicate preference for a coordination-first reading. Three out of four of our successful heuristics use word distribution information generated by Sketch Engine [19] operating on the British National Corpus (BNC). Sketch Engine is an advanced statistical tool supplying information about word distribution, word similarity and word differences. The BNC is a modern corpus containing over 100 million words of English. It is collated from a variety of sources, including some from the same domains as the documents in our corpus. Other researchers have shown that utilising statistical NLP techniques in RE support tools is now a promising area of research [30].

Our system is partially automated. The collection of data from Sketch Engine is by hand, though we are working on creation of an automated collection process. Both the calculation of performance statistics for all the heuristics and the cross-validation exercise are achieved computationally.

The graphs in the following subsections are included to indicate the individual heuristics' performance at various cut-offs. Their performance in these graphs is generally higher than it is when used in the final evaluation because they have not been subjected to cross-validation: there, variables optimised on training data get applied non-optimally to the test data. Precision baselines are shown on all the graphs to indicate how much better the heuristics' performance is than pure chance. (The f-measure baselines are omitted as these are very close to the precision baselines.)

5.1 Coordination-Matching Heuristic

One approach to finding the most likely reading of a coordination is to find out if that coordination occurs frequently in the language. We hypothesise that if a coordination in our dataset is found frequently in a generic corpus, then it is commonly a syntactic unit and a coordination-first reading is the most likely.

We search the BNC for each coordination in our dataset using the word sketch facility in the Sketch Engine, which generates lists of words that are conjoined with and or or. For each coordination, we input one of the two head words and Sketch Engine returns a list of words that it is coordinated with in the BNC. The ranking of the other head word in this list is noted. (A *head word* is the main word of a phrase: the other words in that phrase modify it.) The same procedure must then be performed with the other head word of the coordination, as the difference in overall frequency of the two words can give different rankings. We use as our metric the higher of the two rankings that we have noted.

For the requirement from our dataset:

> Security and Privacy Requirements,

Privacy is ranked 19th for Security, and Security is ranked 8th for Privacy, so 8 is the result of the heuristic on this coordination. Of the 17 survey judges, 4 judged this ambiguity to be ambiguous, 12 judged it to be coordination-first and 1 judged it to be coordination-last. As the number of ambiguous judgements is not larger than the numbers of either of the other two types of judgement, this coordination is not an acknowledged ambiguity. The percentage unacknowledged ambiguity is 1/13 = 7.7%, which is below the average unacknowledged ambiguity (which is 15.3%), so it is not an unacknowledged ambiguity. Neither of the criteria for being nocuous are fulfilled, so this ambiguity is innocuous. For this requirement, this heuristic gives a true positive

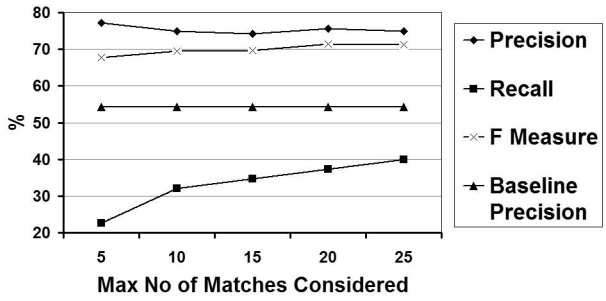

Figure 2. Coordination-matches heuristic

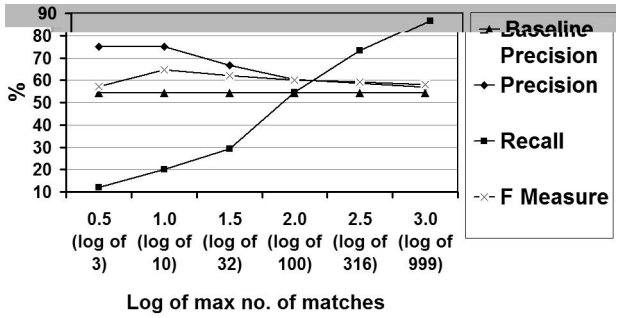

Figure 3. Distributional-similarity heuristic

result for all cut-offs of 8 or greater.

The results that we obtain, using different ranking cut-offs in multiples of 5, are shown in Figure 2. Precision in excess of 21 percentage points above the baseline and 40% recall can be achieved indicating that, even on its own, this heuristic is a useful predictor of innocuous ambiguity. When determining the optimum cut-off for the combined heuristics, we find that the f-measure is optimised when a maximum of 25 matches is used for 7 of the iterations and a maximum of 20 matches is used for the other 3.

5.2 Distributional-Similarity Heuristic

The distributional similarity of two words is a measure of how often those words are found in the same contexts. Distributional similarity is not a measure of the meaning of words: for instance, good and bad have strong distributional similarity even though their meanings are opposite. Our hypothesis for this heuristic, suggested by Kilgarriff [18], is that strong distributional similarity between the head words of coordinated phrases indicates that those phrases form a syntactic unit, indicating a coordination-first reading.

To find the distributional similarity between two head words, we look one of them up in the Sketch Engine's distributional thesaurus, and note the ranking of the other head word in the list of matches that is returned. As with the coordination-matching heuristic, we perform this procedure for each head word of a coordination, using the higher of the two rankings as our metric. In the requirement:

Definition of electrical characteristics and interfaces

no matches between characteristic and interface are found in the thesaurus, and so the heuristic always yields a negative result. Of the 17 survey judges, 9 judged this coordination to be ambiguous, 4 judged it to be coordination-first and 4 judged it to be coordination-last. This means that it is an acknowledged ambiguity. The unacknowledged ambiguity is 4/4 = 100%, so it is also an unacknowledged ambiguity. On both counts this ambiguity is nocuous, and this heuristic will always give a true negative result on this requirement.

Our distributional similarity results are shown in Figure 3. Experimentation has shown us that the predictive power of distributional similarity decreases in a log-linear manner the more matches are considered, so we choose cut-offs accordingly. As can be seen, precision in excess of 20 percentage points above the baseline can be achieved, though recall at this level reaches only 20%. We find that the f-measure is always optimised for the combined heuristics with a maximum of 10 matches, so this is the cut-off we use for all the iterations in the cross-validation exercise.

5.3 Collocation-Frequency Heuristic

This heuristic differs from the previous two in that it predicts coordination-last readings. Here we find out if combinations of the coordinated phrases and modifiers in our dataset are common in the language. We hypothesise that if a modifier is collocated in a corpus much more frequently with the coordinated head word that it is nearest to than it is collocated with the further head word, then it is more likely to form a syntactic unit with only the nearest head word. A coordination-last reading is therefore the most likely.

We find the frequencies in the BNC with which the modifier in each sentence is collocated with the head words of the coordination. Sketch Engine provides lists for most relationships that a word can have with a modifier. We use as our metric the ratio of the collocation frequency with the nearest head word over the collocation frequency with the further head word, with cut-offs set at integer values.

For the requirement from our dataset:

Project manager and designer,

Project has a collocation frequency of 29.55 with manager in the BNC, but it has no collocations there with designer. So the heuristic always yields a positive result for this requirement. 5 survey judges acknowledged this coordination as being ambiguous, 4 judged it to be coordination-first and 8 judged it to be coordination-last. It is not therefore an acknowledged ambiguity. As 4/8 = 50%, which is greater than the average unacknowledged ambiguity, this coordination is however an unacknowledged ambiguity. The heuristic therefore always gives a false positive result.

One aspect of our definition of innocuous ambiguity can

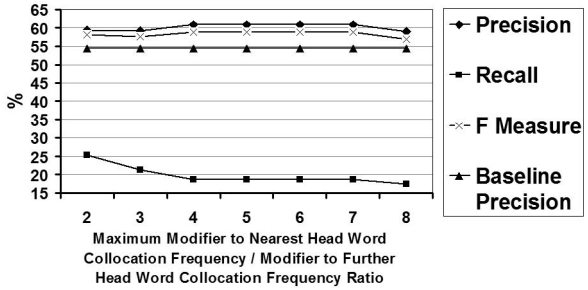

Figure 4. Collocation-frequency heuristic

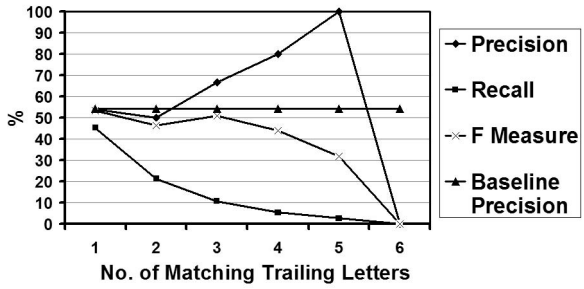

Figure 5. Morphology heuristic

usefully be pointed out here. This is the possibility that a heuristic correctly predicts that an ambiguity is innocuous, but for the wrong reason. For the requirement:

Research and Development Management Information System,

System has a collocation frequency of 14.06 with Development and a collocation frequency of 1.96 with Research. 14.06/1.96 = 7.2, and so the heuristic yields a positive result for this requirement below a cut-off of 8. 4 survey judges judged this coordination to be ambiguous, 12 judged it to be coordination-first, nobody judged it to be coordination-last and 1 judge left no response. It is not therefore an acknowledged ambiguity. Also, as 0/12 = 0, it is not an unacknowledged ambiguity. On this sentence, the heuristic therefore tends to yield a true positive result but for the wrong reason: it thinks it is innocuous because it is coordination-last, but it is innocuous because it is coordination-first.

The results we obtain, using different collocation frequency ratios, are shown in Figure 4. Precision of 6.5 percentage points above the baseline can be achieved, with recall of 18.7%. This performance is modest, but as this heuristic is the only one that successfully predicts coordination-last readings, it is a vital contribution to our combined heuristics' predictive power. When determining the optimum cut-offs for the combined heuristics, we find that the f-measure is always optimised when 4 is the maximum ratio, so this is the cut-off that we use for all the iterations of the cross-validation exercise.

5.4 Morphology Heuristic

This heuristic attempts to capture aspects of the coordinated phrases' morphology in order to predict coordination-first readings. It does not require word distribution information or any other external source of data. The inflectional morphology of a language is the analysis of the changing of words to signify their tense, number, gender etc: in English it consists largely of suffixes such as -ed to indicate past tense, -ing to indicate progressive action, and -s to indicate plurals. The derivational morphology of English is more complex but suffixes, such as -ation and -able, are also very common. We hypothesise that, if the trailing characters of the headwords of the coordinated phrases match, then the words are likely to be of similar types. They would therefore more naturally form a syntactic unit, resulting in preference for a coordination-first interpretation.

To test our hypothesis, we try to match equal numbers of trailing characters from the head words, using the number of characters as our cut-offs. We consider a minimum of one character, capturing most plurals but also a lot of noise, and a maximum of 6 letters, since capturing morphology becomes vanishingly small with consideration of more than a few letters. For the requirement:

It cannot function with the proper installation and configuration,

the trailing characters of installation match those of configuration up to a maximum of 5, so the heuristic gives a positive result for this coordination up to a cut-off of 5. 2 survey judges judged this coordination to be ambiguous, 13 judged it to be coordination-first, nobody judged it to be coordination-last and 2 people entered no response. It is not an acknowledged ambiguity, and as 0/13 = 0 it is not an unacknowledged ambiguity, so the heuristic yields a true positive result on this sentence for all cut-offs up to 5.

The results that we obtain, using different ratios, are shown in Figure 5. Maximum precision is achieved at a cut-off of 5, where it is more than 45 percentage points above the baseline, although recall is only 2.7% here. This shows that this heuristic has only a limited contribution to make to the predictive power of our combined heuristics, but it is a very reliable one. In experimentation to determine the optimum cut-offs for the combined heuristics, we find that the f-measure is optimised when 5 is the cut-off for 9 of the iterations in the cross validation exercise, and 3 is the cut-off for the remaining one.

5.5 Other Heuristics Considered

We experimented using heuristics based on the lengths of the coordinated phrases and based on the number agreement of coordinated nouns. The hypothesis was that disparities in either of these two factors would suggest a lack of similarity, and a coordination-first reading would therefore be less likely. We also tested a simple metric of semantic similarity,

Table 2. Performance of our heuristics

Heuristic (CV = cross-validation exercise)	Recall (%)	Precision (%)	Precision Percentage Points above Baseline	F-Measure $\beta = 0.25$ (%)	F-Measure Percentage Points above Baseline
(Baselines)	100	54.3	-	55.8	-
Coordination-Matches	40.0	75.0	20.7	71.3	15.5
Distributional-Similarity	20.0	75.0	20.7	64.6	8.8
Collocation-Frequency	18.7	60.9	6.6	53.7	-2.1
Morphology	2.7	100	45.7	31.8	-24.0
Heuristics combined (pre CV)	58.7	71.0	16.7	70.1	14.3
Heuristics combined (post CV)	56.3	62.2	7.9	61.6	5.8

based on the closeness of coordinated head words in hierarchies of hypernyms. However, these heuristics demonstrated only very weak predictive power. Whether the modifying phrase appears before or after a coordination appears to have very little bearing on the preferred reading of a coordination. Also, the part of speech of the coordinated head words and the part of speech of the modifier also have no predictive effect for our dataset.

6 Combined Heuristics

Working on research in some ways parallel to ours, Natt och Dag et al. advocate aggregating disparate similarity calculation techniques in order to improve performance [22]. We combine our heuristics using disjunction: the combined heuristics are considered to give a positive result if any one of the individual heuristics gives a positive result. Table 2 presents the results of the heuristics combined both before and after the cross-validation exercise. Combining the heuristics using logistic regression [21], a technique much used in medicine to determine the predictive accuracy of diagnostics, gives us very similar results. The individual heuristics' results are also presented in Table 2, without having been subject to cross-validation, at the cut-off most commonly used for them.

As can be seen, all our individual heuristics have much higher precision than the baseline. The precision baseline is calculated simply by assuming that all the coordinations are innocuous. As this is an assumption that we in no way want to make, because it would allow many nocuous ambiguities to pass unnoticed, it is appropriate that the precision of the individual heuristics is high at the expense of low recall. Combining the heuristics increases the recall considerably, indicating that their coverage of innocuous ambiguities is to some extent complementary. However, the precision of the combined heuristics is less than the precision of most of the heuristics. This indicates that the intersection of the coverages of the individual heuristics tends to occur when true positives are achieved. This decreases the contribution of these true positives to the combined heuristics' prediction.

When applying the 10-fold cross-validation exercise to the data, and adjusting the cut-offs to maximise the f-measure for the combined heuristics, our performance drops markedly. This is because the cross-validation exercise is using non-optimal cut-offs for its calculations on the test data. However, we still achieve 7.9 percentage points precision over the baseline, and with a much higher recall than any of the individual heuristics.

7 Threats to Validity

Here we discuss some ideas which might invalidate our belief that our approach is suitable for tackling ambiguity in RE documents.

Context can have a major disambiguating influence. Although easily appreciated by humans, it is notoriously difficult for computers to capture and interpret the large amounts of contextual data necessary for interpreting ambiguities. Also, our judges cannot afford to spend much time on the task, so we do not show them any context beyond each sentence containing a coordination. The requirements in our dataset may therefore have been judged to be more ambiguous than they actually are. However, some of the texts from which they are taken present real coordination ambiguities even when context is considered. Perhaps the best example we have, though it is a complex one, is:

> Benefits associated with the use of XXX include improved design quality and decreased construction and operations cost and time

where the coordination ambiguity becomes compounded and many different structures and readings are possible.

Our approach might be invalidated by using other more established techniques to solve the problem effectively. However, many of these fall into the category of techniques which requirements engineers are reluctant to use, as discussed in Section 2. Good training in RE practice and terminology would make stakeholders better informed, and of the same mind, when writing and interpreting RE documents. Proficiency of all stakeholders in the language used would prevent some nocuous ambiguity: some of the documents

we collected are written in surprisingly poor English.

Using specific RE corpora, or large corpora of computing documents, might improve the heuristics' performance, but we are not aware of suitable corpora of these types.

The idea that "an ambiguity detection technique needs to be tailored to an RE context in order to be most effective" [17] contains some truth, but it mainly applies to domain-specific ambiguities and not to the type that we consider.

8 Conclusions

Our results show that our heuristics correctly identify a considerable percentage of requirements which are judged to contain innocuous coordination ambiguity. Requirements engineers then have the simplified task of understanding the remaining nocuous ambiguities and rewriting them in a less ambiguous form. We would still prefer the precision of our combined heuristics to be higher, possibly at the expense of recall: judging nocuous ambiguities to be innocuous is dangerous, whereas including some innocuous ambiguities with the nocuous ones is merely time wasting.

From the performance of our coordination-matches heuristic we conclude that a surprising number of coordinations found in specialised requirements documents are also found in a generic corpus. From the performances of our distribution-similarity heuristic and our collocation-frequency heuristic we conclude that a large number of words that are found together in other relationships in those documents are also found to have distributional relationships in a generic corpus. This encourages us to believe that the types of data that we attempt to match are appropriate, and that in the absence of more domain-specific corpora a large generic corpus can be very effective.

The precision of our morphology heuristic indicates that matching the morphology of the head words of a coordination is a very accurate way of predicting some coordination-first readings. This may be particularly true in RE as many words are technical, and such words are often developed from their base forms using the same linguistic processes. For instance, the coordinated words installation and configuration, introduced previously, both describe the resultant state of processes. It might be expected that such words with similar functions are likely to form a syntactic unit in a coordination, and their morphology reflects this.

The performance of the combined heuristics shows that recall can be greatly improved at only a small loss of precision. However, there is scope for improvement. The results after implementation of the cross-validation exercise demonstrate the danger of overfitting the data, and the results are penalised because of this. We feel sure that using a larger number of sentences would improve our combined heuristics' performance by limiting this effect.

We have shown that, even for a relatively small corpus and for a single type of linguistic construction, a substantial amount of nocuous ambiguity can be found in requirements: using the evaluation method presented here, we find it in 2.6% of the sentences in our corpus. We also find that people's judgements can vary quite widely, demonstrated by the average unacknowledged ambiguity of 15.3% over all the sentences in our dataset. This represents a large potential problem, especially as some sentences have an unacknowledged ambiguity that is much higher than this average.

9 Future Work

Two new heuristics in particular may offer us increased performance. The technical terminology found in requirements documents appears to have high morphological complexity, so to exploit this further we are developing a heuristic that we hope will have increased recall by using greater intelligence about the morphology it is capturing. Secondly, we are evaluating the effectiveness of hybrid heuristics that combine the complementary qualities of semantic similarity and distributional similarity.

We believe our technique is scalable: our model of ambiguity and use of large corpora would be just as effective when applied to large volumes of requirements documents as it is when applied to smaller datasets. Also, we intend our approach to be extensible so that it can deal with other types of ambiguity. The most obvious types to address next are prepositional phrase attachment and noun compounding: these are common, and tractable using the Sketch Engine and the BNC. As the nocuous/innocuous distinction can be applied to *all* ambiguities, we also propose using it to address other types known to be particularly problematic in requirements, for instance quantifiers and negations [4]. Our ultimate aim is to create a tool for requirements engineers that acts like a *wizard* in conjunction with a word processor, distinguishing between nocuous and innocuous ambiguities of many different types.

Acknowledgements

The authors would like to thank Alistair Sutcliffe and the anonymous referees for their helpful feedback on earlier drafts of this paper.

References

[1] C. B. Achour. Guiding scenario authoring. In G. Grosz, editor, *Proceedings of the 8th European Japanese Conference on Information Modelling and Knowledge Bases*, pages 152–171, 1998.

[2] V. Ambriola and V. Gervasi. Processing natural language requirements. In *Proc. of the 12th International Conference on Automated Software Engineering*, pages 36–45, Los Alamitos, CA, U.S.A., 1997. IEEE Computer Society Press.

[3] D. Berry and E. Kamsties. The syntactically dangerous all and plural in specifications. *IEEE Software*, 22(1):55–57, Jan/Feb 2005.

[4] D. M. Berry, E. Kamsties, and M. M. Krieger. From contract drafting to software specification: Linguistic sources of ambiguity, 2003. A Handbook.

[5] B. W. Boehm. *Software Engineering Economics*. Prentice-Hall, Englewood Cliffs, NJ, U.S.A., 1981.

[6] S. Boyd, D. Zowghi, and A. Farroukh. Measuring the expressiveness of a constrained natural language: An empirical study. In *Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE'05)*, pages 339–352, Washington, DC, USA, 2005. IEEE Computer Society.

[7] F. Chantree, A. Kilgarriff, A. de Roeck, and A. Willis. Disambiguating coordinations using word distribution information. In *Proceedings of Recent Advances in Natural Language Processing (RANLP)*, Borovets, Bulgaria, 2005.

[8] F. Chantree, A. Kilgarriff, A. de Roeck, and A. Willis. Using a distributional thesaurus to resolve coordination ambiguities. Technical Report 2005/02, The Open University, Milton Keynes, U.K., 2005.

[9] D. P. Freedman and G. M. Weinberg. *Handbook of Walkthroughs, Inspections, and Technical Reviews: Evaluating Programs, Projects, and Products*. Dorset House Publishing Co., Inc., New York, NY, USA, 2000.

[10] N. Fuchs and R. Schwitter. Attempto controlled english (ace). In *Proceedings of the first international workshop on controlled language applications*, 1996.

[11] D. C. Gause and G. M. Weinberg. *Exploring requirements: quality before design*. Dorset House, New York, 1989.

[12] V. Gervasi and B. Nuseibeh. Lightweight validation of natural language requirements: a case study. In *Proceedings of the 4th IEEE International Conference on Requirements Engineering*. IEEE Computer Society Press, 2000.

[13] V. Gervasi and B. Nuseibeh. Lightweight validation of natural language requirements. *Software Practice and Experience*, 32(2):113–133, 2002.

[14] L. Goldin and D. M. Berry. Abstfinder, a prototype natural language text abstraction finder for use in requirements elicitation. *Automated Software Engineering*, 4(4):375–412, October 1997.

[15] M. J. Hillelsohn. Better communication through better requirements. *Crosstalk: The Journal of Defense Software Engineering*, 17(4), April 2004.

[16] E. Kamsties. *Surfacing Ambiguity Natural Language Requirements*. PhD thesis, Fraunhofer IESE, Kaiserslautern, Germany, 2001.

[17] E. Kamsties, D. Berry, and B. Paech. Detecting ambiguities in requirements documents using inspections. In M. Lawford and D. L. Parnas, editors, *Proceedings of the First Workshop on Inspection in Software Engineering (WISE'01)*, pages 68–80, 2001.

[18] A. Kilgarriff. Thesauruses for natural language processing. In *Proc. of NLP-KE*, pages 5–13, Beijing, China, 2003.

[19] A. Kilgarriff, P. Rychly, P. Smrz, and D. Tugwell. The sketch engine. In *Proc. of EURALEX 2004*, pages 105–116, 2004.

[20] B. L. Kovitz. *Practical Software Requirements: A Manual of Content & Style*. Manning Publications, Greenwich, CT, USA, 1999.

[21] N. Landwehr, M. Hall, and E. Frank. Logistic model trees. In N. Lavrac, D. Gamberger, L. Todorovski, and H. Blockeel, editors, *Proceedings of the 14th European Conference on Machine Learning*, pages 241–252. Springer, 2003.

[22] J. Natt och Dag, B. Regnell, V. Gervasi, and S. Brinkkemper. A linguistic-engineering approach to large-scale requirements management. *IEEE Software*, 22(1):32–39, Jan/Feb 2005.

[23] A. A. Porter, L. G. Votta, and V. R. Basili. Comparing detection methods for software requirements inspections: A replicated experiment. *IEEE Transactions on Software Engineering*, 21(6):563–575, June 1995.

[24] R. Quirk, S. Greenbaum, G. Leech, and J. Svartvik. *A Comprehensive Grammar of the English Language*. Longman, New York, 1985.

[25] P. Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130, 1999.

[26] H. B. Reubenstein and R. C. Waters. The requirements apprentice: An initial scenario. In S. Greenspan, editor, *Proceedings of the 5th International Workshop on Software Specification and Design*, pages 211–218, Pittsburgh, PA, U.S.A., 1989. IEEE Computer Society Press.

[27] C. Rupp. Linguistic methods of requirements engineering (nlp). In *Proceedings of European Software Process Improvement (EuroSPI 2000)*, Copenhagen, Denmark, 2000.

[28] K. Ryan. The role of natural language in requirements engineering". In *In Proceedings of the IEEE Int. Symposium on Requirements Engineering*, pages 240–242. IEEE Computer Society Press, 1993.

[29] J. Ryser and M. Glinz. Scent - a method employing scenarios to systematically derive test cases for system test. Technical Report 2000.03, Institut für Informatik, University of Zurich, Switzerland, 2000.

[30] P. Sawyer, P. Rayson, and K. Cosh. Shallow knowledge as an aid to deep understanding in early phase requirements engineering. *IEEE Transactions On Software Engineering*, 31(11):969–981, 2005.

[31] F. Shull, I. Rus, and V. Basili. How perspective-based reading can improve requirements inspections. *IEEE Computer*, 33(7), July 2000.

[32] L. M. Solan. *The Language of Judges*. University of Chicago Press, Chicago, U.S.A., 1993.

[33] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, U.K., 1979.

[34] T. Wasow, A. Perfors, and D. Beaver. The puzzle of ambiguity. In O. Orgun and P. Sells, editors, *Morphology and the Web of Grammar: Essays in Memory of Steven G. Lapointe*. CSLI Publications, 2003.

[35] S. M. Weiss and C. A. Kulikowski. *Computer systems that learn: classification and prediction methods from statistics, neural nets, machine learning, and expert systems*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1991.

[36] D. Zowghi, V. Gervasi, and A. McRae. Using default reasoning to discover inconsistencies in natural language requirements. In *Proceedings of the 8th Asia-Pacific Software Engineering Conference*, Macau, China, 2001.