ISSN 1744-1986

Technical Report N^{o} 2005/03

Nocuous Ambiguities in Requirements Specifications

Francis Chantree
Bashar Nuseibeh
Anne De Roeck
Alistair Willis

3rd March 2005

Department of Computing
Faculty of Mathematics and Computing
The Open University
Walton Hall,
Milton Keynes
MK7 6AA
United Kingdom

http://computing.open.ac.uk

Nocuous Ambiguities in Requirements Specifications

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

Abstract

In this paper we present a novel approach that automatically alerts authors of requirements specifications to the presence of potentially dangerous ambiguities in their text. We first establish the notion of "nocuous" ambiguities, i.e. those that are likely to lead to misunderstandings. We focus on coordination ambiguity, which occurs when words such as "and" and "or" are used. Our starting point is a dataset of ambiguous phrases from a corpus of requirements specifications, and a collection of associated human judgements about their interpretation. We then use machine learning techniques combined with syntactic, semantic and word distribution heuristics to eliminate instances of text which people interpret easily. We report on a series of experiments and evaluate the performance of our approach against the collection of human judgements. Our machine learning algorithm has an accuracy of 75% compared to a 59.6% baseline.

1. Introduction

Ambiguity – the capability of a text to be interpreted in more than one way – is endemic in natural language. It is therefore also a considerable problem for requirements that are written in natural language [4] [15]. Gause and Weinberg observe that ambiguity is "the fundamental problem of requirements definition" [11], and Berry et al. suggest that unintended ambiguity is the "Achilles' heel" of software requirements specifications [4]. If stakeholders interpret a requirement in different ways, this can result in an incorrect implementation. Ambiguity can be more intractable than other defects, such as incompleteness, and research has shown that it more frequently results in misunderstandings [16].

Our work is driven by the desire to eliminate ambiguities that will lead to misunderstandings, and therefore to faulty implementations. Carrying this out 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 [6]. However, locating errors is nontrivial, and even requirements specifications that have been checked many times can still contain defects [12].

As well as ambiguities that lead to misunderstandings, there are others which have only one obvious interpretation. We refer to the former as *nocuous* ambiguities and to the latter as *innocuous* ambiguities. Nocuous ambiguities can be both those that are 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 specification writers automatically identify innocuous ambiguities. These can then be eliminated from further analysis, leaving the remaining nocuous ambiguities to be disambiguated at a future stage. Natt och Dag et al. cite industrial experience that motivates the need for such automated support for requirements management [23], and Gervasi and Nuseibeh suggest that this is both feasible and useful [12].

To test our approach, we look at ambiguity arising from coordinations, which are recognised as "a pernicious source of structural ambiguity in English" [27]. *Coordination ambiguity* is a structural (i.e. syntactic) ambiguity: alternative readings result from the different ways in which a sequence of words containing a conjunction can be grammatically structured. For example, in the sentence:

Patients conditions are inspected and recorded automatically

the coordination ambiguity is nocuous: *automatically* could just as easily apply to both *inspected* and *recorded* as to just *recorded*, and a misunderstanding might occur because of this.

Previous work on ambiguity in the requirements engineering (RE) literature [16] [4] [17] [28], has tended to focus on specifically RE-related issues. This is understandable of course, but it means that some types of ambiguity are not dealt with rigorously. 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 by developing and adapting techniques from natural language processing (NLP) to identify coordination ambiguities in requirements. The paper therefore bridges the two disciplines of RE and NLP.

Our approach is to gather a dataset of sentences containing coordinations from a corpus of requirements specifications, and determine how nocuous the ambiguities are by means of surveys. We then use heuristics, utilising disparate types of linguistic information about coordinations, to predict whether they are nocuous or innocuous ambiguities. Then we apply a machine learning (ML) algorithm to the data created by the heuristics, in order to determine the combined accuracy of the heuristics.

The paper is structured as follows. In Section 2 we provide some background for our research, including a discussion of work on inspection techniques to which our approach bears some similarity. In Section 3 we describe our treatment of coordination ambiguity in RE. Section 4 describes how our training data is created: identifying coordination ambiguities in a corpus of requirements specifications, and then having a group of human participants judge how nocuous they are. Section 5 presents four heuristics that we developed to predict innocuous ambiguity. Section 6 describes the ML algorithm that is applied to the results of the heuristics. Sections 7 and 8, respectively, conclude the paper and discuss future work. All the examples used are taken from our corpus of requirements specifications.

2. Background

Ambiguity is *avoided* in textual requirements specifications by use of techniques such as controlled languages, [10] [2], and style guides, [20] [30]. Alternatively, ambiguity is *detected* in requirements specifications by use of techniques such as inspections, fit criteria, and test cases. However, Kamsties observes that fit criteria and test cases generally only reduce vagueness and generality, and that requirements authors are sometimes reluctant to use controlled languages and style guides [16]. (Generality [13] and vagueness [14] are linguistic concepts similar to ambiguity, but they cause misunderstandings due to lack of specificity rather than to widely different meanings.)

2.1 Inspection Techniques

Inspection techniques are known to be effective and efficient ways of reducing errors in requirements documents [31]. Some approaches use sets of heuristics to detect ambiguity [29] [11], and the ambiguity poll technique described in the last of these demonstrates the difficulty in obtaining consensuses of opinion about ambiguity. Inspection techniques have often been used to detect inconsistency and incompleteness in requirements specifications, [36] [25]. However, little work has been done on applying them to detect ambiguities in requirements specifications. Kamsties et al. have 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 inspection approaches, both of the defect-based reading [25] and the perspective-based reading [31] varieties. The questions posed in these approaches bring ambiguity to the readers’ attention, but will not necessarily make them aware of the extent to which misinterpretations might occur.

One study that does investigate ambiguity in requirements documents more thoroughly using inspections is [17]. However, this work does not deal explicitly with structural ambiguities like coordination ambiguity. Coordination ambiguity has in fact received surprisingly little attention in either the NLP or the RE literatures.

3. The Ambiguity Problem

Coordination ambiguity can occur whenever a coordinating conjunction is used. In this paper we have chosen to look only at the coordinating conjunctions *and*, *or*, and *and/or*. *And* and *or* are the most widespread coordinating conjunctions, accounting for 97% of all such words in a recent study [24]. In the requirement:

Display categorized instructions and documentation,

it is unclear what is categorized. 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 categorized. We call this a *coordination last* reading. This will affect what type of display is implemented as a result of this requirement.

3.1 Nocuous and Innocuous Ambiguity

Ambiguity in language is often not a problem for humans. This is because humans are capable of using their knowledge of the world, and the context supplied by the words that surround an ambiguity, to disambiguate that ambiguity. For instance, in:

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

a coordination first reading in this context is the most likely, with *vector-based* applying to both *inputs* and *outputs*. 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.

We say that ambiguities which people read in different ways have *multiple readings*, and are *nocuous ambiguities*. The sentence given in the introduction is a clear example of the multiple reading situation: both coordination first and coordination last readings are likely.

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* referred to are of the *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 of the stakeholders realises the potential ambiguity, they can discuss what the most likely meaning is or contact the author to elicit the intended meaning. The ambiguous passage of text can then be rewritten in a less ambiguous form and incorrect implementation can be averted.

However, two or more readers can interpret a passage of text in different ways and each assume that their own interpretation is the only obvious one [5]. We call this phenomenon *unacknowledged ambiguity*. For instance, in:

> The user may define architectural components and connectors,

some might consider that *architectural* applies to both *components* and *connectors*, whereas others might consider that it only applies to *components*. If there are many types of *components* and *connectors*, then this unacknowledged ambiguity may have consequences when the requirements are implemented. There is a risk that none of the stakeholders will recognise that this sentence is an acknowledged ambiguity, but will interpret it differently. In that case, 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*, which is one of Gause's five most important sources of requirements failure [4]. The problem becomes even more likely when the author and the stakeholders have different language abilities; e.g., if one or both of them is not a native speaker of the language being used.

Both acknowledged and unacknowledged ambiguities are nocuous because multiple readings are possible. Figure 1 shows the relationships between the types of ambiguity that we have been discussing. The RE community has recognised that unacknowledged ambiguities are a serious problem [17] [16], and that errors resulting from unconscious misunderstandings are surprisingly common [3].

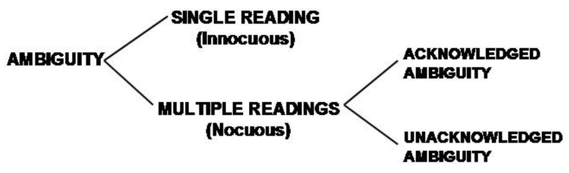

Figure 1. Nocuous/innocuous and acknowledged/ unacknowledged Ambiguity

3.3 Notification rather than Disambiguation

Many NLP researchers have worked on resolving ambiguities to aid the understanding, generating, or translating of text. This has never met with complete success, due to the inability of computers to capture all the domain knowledge necessary to disambiguate language. Our approach is therefore to notify users of potential ambiguities and to then leave them to perform disambiguation. This utilises the proficiency of computers at finding potential ambiguities automatically, and the requirements engineers' ability to decide how to rewrite them less ambiguously.

4. Creating Training Data

Our training data is created by locating sentences containing coordinations in a corpus of requirements specifications. Each coordination is potentially ambiguous. However, there are no absolute criteria for judging ambiguity, as it is a product of the meanings that people that assign to language [33]. Therefore, we capture human judgements to determine the most likely readings of the ambiguities in our sentences. We obtain these judgements by surveying the opinions of a group of computing professionals. Rather than rely upon the judgement of one participant, we take a consensus of opinions from multiple participants. Such an approach is known to be very effective albeit quite expensive [4].

4.1 Obtaining Example Requirements from a Corpus of Requirements Specifications

We have built (and are continually extending) a corpus of requirements specifications. In this corpus we identify sentences - and titles, bullet points, etc - that contain conjunctions. We eliminate sentences which can give only one syntactic reading. Other sentences are included in our surveys, regardless of how obvious one particular reading might be. Such a lack of discrimination is necessary in order to create realistic training data. After the elimination process, the data used for our surveys consisted of 52 different sentences, each containing a coordination ambiguity. Each sentence represents a requirement.

4.2 The Ambiguity Surveys

The 52 requirements obtained from our corpus were shown to 17 participants in 2 separate surveys. The participants were asked to judge whether each coordination was coordination first, coordination last, or ambiguous so that it might lead to misunderstanding. In the last case, the coordinated expression is then classed as an acknowledged ambiguity for that participant. Clearly there is an elusive dividing line between what would and what would not lead to misunderstandings. We take the view that, by using a large number of participants, we obtain a reliable consensus about where this line lies for each example. The list of requirements that made up the ambiguity surveys is given in the appendix to this paper.

4.3 Identifying Requirements that contain Nocuous Coordination Ambiguities

We say that a requirement contains acknowledged coordination ambiguity if it has been judged to be ambiguous by the survey participants at least as often as it has been judged by them to be coordination first and at least as often as it has been judged by them to be coordination last.

A requirement is judged to contain an unacknowledged coordination ambiguity if it contains an above average percentage unacknowledged ambiguity. This is calculated based solely on the numbers of non-ambiguous judgements: i.e. the judgements of coordination first and coordination last.

$$\text{Percentage Unack-nowledged Ambiguity} = \frac{\text{Numbers of the least popular non-ambiguous judgement}}{\text{Numbers of all the non-ambiguous judgements}}$$

The appendix shows whether the requirements are judged to contain acknowledged and unacknowledged ambiguities.

5. Heuristics for Predicting Innocuous Ambiguity

We now describe the heuristics which we use to predict innocuous ambiguity. Each heuristic has some ability to predict which coordination ambiguities are easily disambiguated by humans. We hypothesise that, because the heuristics measure different types of information, they may be complementary. Used as a set, they will have a coverage which is the union of the individual heuristics' coverages. Where the coverages of the individual heuristics overlap, it is to our advantage if they agree with one another. Indeed, Natt och Dag et al. [23], suggest that an aggregation of several different similarity calculation techniques might improve precision.

We use measures of precision, recall, and f-measure. F-measure is a metric which combines precision and recall. True positives in this study are requirements which are judged to be innocuous ambiguities by our survey participants and for which a heuristic yields a positive result.

$$\text{Precision} = \frac{\text{Number of true positives}}{\text{Total number of requirements where the heuristic gives a positive result}}$$

$$\text{Recall} = \frac{\text{Number of true positives}}{\text{Total number of requirements judged to contain innocuous coordination ambiguity by our participants}}$$

$$\text{F Measure} = \frac{1}{\dfrac{\alpha}{\text{precision}} + \dfrac{(1-\alpha)}{\text{recall}}}$$

We use a weighting value of $\alpha = 0.9$ when calculating the f-measure, which weights the f-measure strongly in favour of precision. This is done because we want each heuristic to be a precise indicator of lack of ambiguity, and we hope that good recall will be achieved by using all the heuristics together. The f-measure metric is used to determine which cut-off point we use for each heuristic.

5.1 Distributional Similarity

Distributional similarity is the extent to which words are found in similar contexts to one another. Distributional similarity is not a measure of the meaning of words: for instance, *good* and *bad* have strong distributional similarity to each other because they are often found in the same contexts, even though their meanings are opposite. Our hypothesis for this heuristic is that strong distributional similarity of coordinated head words indicates that a coordination first reading of the coordination is most likely. This idea is suggested in [18]. (A head word is the main word of a phrase: the other words in that phrase modify it.)

To find the distributional similarity between two words we look them up in the distributional thesaurus provided with the Sketch Engine [19]. This thesaurus returns a list of the words with the most distributional similarity, up to a cut-off limit which one enters. We use cut-off points that give an even distribution on a log scale. All words entered must be lemmatised, i.e. the base forms of the words must be used. Verbs, nouns and adjectives can be entered. The Sketch Engine calculates distributional similarity using a similarity measure [21]. We use only the rankings of the matches given by the similarity measure, as these are considered to be a more useful metric than the similarity measure figures themselves [22].

However, instead of simply entering the coordinated headwords in the thesaurus, we employ a modification in order to improve our performance. We attempt to match the underlying verbs of the coordinated words, where possible. This is possible in 69% of the requirements sentences. The hypothesis is that morphologically developed nouns, such as *operating* and *performance,* in:

Operating and performance requirements.

might be comparatively rare due to the fact that several such words with similar meanings tend to evolve in a language. Such sparseness of data would produce unreliable lists of matches in the thesaurus. Therefore we match the underlying verbs *operate* or *perform,* which might exhibit less sparseness. In this case, the underlying verbs do have greater distributional similarity than the original words: *operate* is the 5th highest match for *perform.* Where no underlying verbs can be found, we use a back-off procedure of reverting back to matching the original words. The phrase given as an example in this section was judged strongly to have a coordination first reading in our surveys.

The distributional similarity results that we achieved are shown in Figure 2. The f-measure peaks when a maximum of 32 matches is used, so that is the cut-off point we use for all our distributional similarity analysis. The performance of the f-measure at this point is 8.6% above the f-measure baseline.

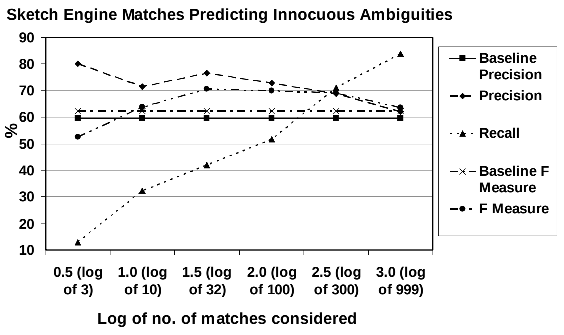

Figure 2. Distributional similarity results

5.2 Semantic Similarity

Semantic similarity is the extent to which the meanings of words are similar. Our hypothesis for this heuristic is that strong semantic similarity of coordinated head words indicates that a coordination first reading of the coordination is most likely. This idea is suggested, and investigated, in [27]. There is some evidence that distributional similarity and semantic similarity provide complementary views of similarity [7].

We determine semantic similarity using a simple distance metric based on edge counting [26]. We look up the coordinated head words in WordNet[1] and generate the hypernym lists. (A hypernym is a word that is more generic than a given word and whose meaning subsumes the meaning of that word: e.g. *animal* is a hypernym of *cat.*) These lists give the different meanings of a word, and the hierarchy of the concepts that subsume each meaning. We locate the concepts which appear in the hypernym lists of both words, i.e. the ones that are subsuming concepts of both the coordinated head words. A link between two concepts or between a concept and a word meaning is an *edge*. We determine which of the subsuming concepts has the least combined number of edges from the word meanings. We call this concept the *lowest common ancestor*. Where no lowest common ancestor is found, we create a dummy *thing* concept which subsumes the topmost concepts of all the hierarchies [27].

For the requirement:

Facilitate the scheduling and performing of works

the lowest common ancestor of the coordinated words is *activity*. This is three levels up the hierarchy from *scheduling* (via, *planning* and *preparation/readying*),

[1] (version 2) http://www.cogsci.princeton.edu/cgi-bin/webwn

and one level up from *performing*. The combined distance that it has from the word meanings is therefore 4 edges, so 4 is the measure that we use as the semantic similarity of these two words. This sentence was judged strongly to have a coordination first reading in our surveys.

The semantic similarity results that we achieved are shown in Figure 3. The f-measure is highest when maximums of 5 and 10 edges are considered, and we use the latter cut-off point for all our semantic similarity analysis. The f-measure at this point is 2.7% above the f-measure baseline.

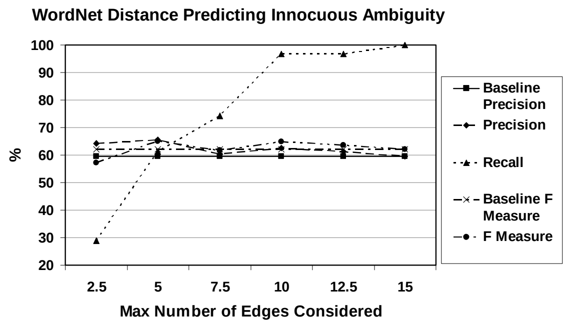

Figure 3. Semantic similarity results

5.3 Noun Number

Coordinations of nouns make up 86% of our dataset. Whether a noun is singular or plural is termed its *number*. A noun coordination can be of two singular nouns, two plurals, or one of each. Our hypothesis is that a coordination first reading is less likely if the coordination is of one of each. This is because dissimilar items will be less likely to form semantic units. The importance of number agreement when determining how a noun coordination should be read is discussed in [27].

In the requirement:

It targeted the project and election managers,

the numbers of the coordinated nouns are both singular. It would read much less naturally as a coordination first interpretation if one of the coordinated words was a plural. This sentence was judged strongly to have a coordination first reading in our surveys.

We find that coordinations of singular nouns and coordinations of plural nouns exhibit similar characteristics, indicating innocuous ambiguity. To improve recall it is advantageous to indicate innocuous ambiguity by using either both singular or both plural as the predictor. The number agreement results that we achieved are shown in Figure 4. The highest point of the f-measure is when the coordinated nouns are either both singular or both plural. The f-measure at this point is 5.7% above the f-measure baseline.

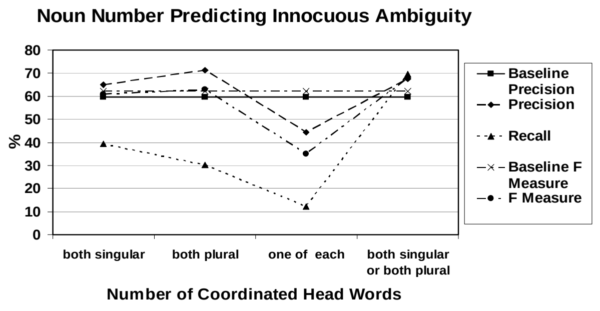

Figure 4. Noun number agreement

5.4 Phrase Length Difference

Prosody (the rhythmic and intonational aspects of language) can influence how ambiguities are resolved. Sentences are read in *chunks*, which represent how words are grouped together in a reader's mind [1]. This can therefore influence how coordinations are read. Our hypothesis is that if the coordinated phrases are equal in length, then a coordination first reading will be the most likely. This is because readers will be more inclined to consider chunks of the same length, as opposed to those of dissimilar lengths, to be equal in status when processing the structure of the sentence.

In the requirement:

Minimize the time span and human resources of regular inspection,

both of the coordinated phrases have a length of 2, and the sentence was judged quite strongly to have a coordination first reading.

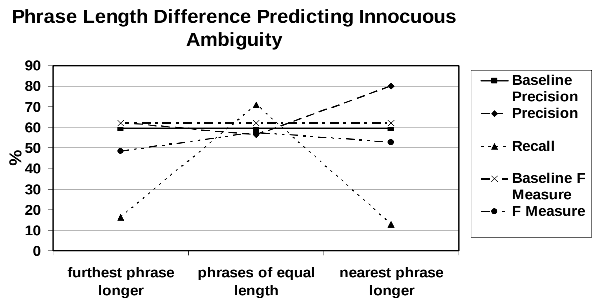

Figure 5. Difference in phrase lengths

The results that we achieved are shown in Figure 5. Equal phrase length is only a weak indicator of innocuous ambiguity. The f-measure at no point rises above the f-measure baseline, but we nevertheless choose to use this as an indicator as the recall is high here. The f-measure when phrase lengths are equal is 4.5% below the f-measure baseline.

5.5 Other Heuristics Considered

We considered using other heuristics, in addition to those already mentioned, utilising lexical information. These were: the part of speech of the external modifier, the part of speech of the head words, and the conjunction itself. However, we found that none of these were predictive of innocuous coordination ambiguity, so they are not used in our analysis nor elaborated any further in this paper.

6. Combining Predictions Using ML

ML has been used for various RE applications such as traceability management [32], theory refinement [35], and requirements evolution [9]. It has not, to our knowledge, been used to manage requirements ambiguities as presented in this paper.

To combine the predictive effects of our heuristics we use the TiMBL ML algorithm [8]. TiMBL is a memory-based learning method, based on the k-nearest neighbour approach, and has often been used for NLP applications. The algorithm we use within TiMBL is IB1, which tends to give greater accuracy at the expense of speed [8], and we look at only one nearest neighbour. We use the leave-one-out method of cross-validation: each line of data is considered in turn to be the test item and is tested against all the remaining lines. This is an elegant way of getting top quality, and almost unbiased, results from small data samples [34]. Each feature in the ML algorithm is a heuristic and its associated data.

The result of running this ML algorithm on our data is that 39 out of the 52 sentences were judged correctly, both nocuous and innocuous, which represents 75% accuracy. This is 15.4% above the baseline accuracy of 59.6% if all sentences are assumed to contain innocuous coordination ambiguities. Considering only prediction of innocuous ambiguities, the precision and recall figures are 71.4% and 96.8% respectively. These results are given in the last line of Table 1; the baseline figures for the precision and recall of predicting innocuous ambiguities are given in the first line.

Table 1 also lists statistics generated by the ML algorithm for each feature, followed by a recapitulation of the individual heuristics' performances. The information gain statistic is a measure of how much each individual feature contributes to our knowledge of the correct class label [8]. The highest information gain is for the distributional similarity heuristic, indicating that this is the most useful. This is in line with the individual heuristics' results. The fact that semantic similarity produces a higher information gain than noun number, despite having a lower precision and f-measure, is due to the very high recall obtained at the cut-off point we use.

It is interesting to note that if we add in the lexical heuristics described in Section 5.5, the overall accuracy of our ML algorithm, predicting both nocuous and innocuous ambiguities, drops to 63.5%.

Table 1. Comparison of performances

Predictor	Info Gain	Prec-ision	Recall	F-Measure ($\alpha = 0.9$)
Baseline		59.6	100	62.1
DS	0.0431	76.5	41.9	70.7
SS	0.0297	62.5	96.8	64.8
NN	0.0152	67.6	69.7	67.8
PLD	0.0158	56.4	71.0	57.6
ML		71.4	96.8	73.3

7. Conclusions

Our results show that our set of heuristics correctly identifies a considerable percentage of requirements sentences which contain innocuous coordination ambiguity. The requirements engineer then has the simplified task of rewriting the remaining nocuous ambiguities in a less ambiguous form.

We found that all our heuristics, even phrase length difference, added to the performance of our ML algorithm. The fact that the ML algorithm's f-measure is higher than the f-measures of all the individual heuristics implies that our heuristics are indeed complementary. The fact that the ML algorithm's recall is no higher than that of the heuristic with the highest recall, semantic similarity, yet its precision and f-measure is much better, means that different results are being recalled in that case. This shows that the ML algorithm is successfully combining the coverages of the individual heuristics. However, we would prefer our precision to be higher 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.

The distributional similarity heuristic has the best precision, and its f-measure score is not far short of that of the ML algorithm. This indicates either that distributional similarity is inherently the most accurate indicator of innocuous ambiguity, or that the distributional similarity heuristic is simply better developed than the others. The former conclusion implies that using other heuristics, in combination with the distributional similarity heuristic, might give even better results. The latter conclusion implies that the other heuristics need to be developed further. Our corpus of requirements specifications reveals a considerable number of coordinations of technical words which have high distributional similarity, see examples in Table 2. We believe this to be a significant feature of the genre.

Table 2. Coordinated head words displaying high distributional similarity

1st Head Word	2nd Head Word	Ranking of Match
input	output	1
hardware	software	1
implement	execute	1
design	develop	3
operate	perform	5
item	component	6
date	location	8
record	history	9
fund	resource	10

We have found that people's perceptions of ambiguity can vary widely: for instance, over a quarter of our requirements sentences produced a percentage unacknowledged ambiguity of over 20%. This confirms our concern about the prevalence of unacknowledged ambiguity in requirements specifications.

8. Future Work

There are other heuristics which could be added to our set. For instance, it would be interesting to look at the distributional relationships between the coordinated phrases and the external modifiers. Information of this type is available in Sketch Engine, and we intend to use this in future research. We also intend to develop an improved version of our semantic similarity heuristic which links the word meanings with greater precision. We can then further test the hypothesis that complementary results using different similarity measures can be successfully applied to ambiguity in RE.

We intend to expand our approach to deal with other types of ambiguity. Our ultimate aim is to create an ambiguity notification tool for requirements engineers that acts like a wizard, distinguishing between nocuous and innocuous ambiguities of several different types. We believe that our technique is scalable, and could be applied to large requirements documents, though it would take time to train it to be a reliable predictor over a wide range of ambiguities.

References

[1] Abney, S., “Chunks and Dependencies: Bringing Processing Evidence to Bear on Syntax”, In *Computational Linguistics and the Foundations of Linguistic Theory,* CSLI, 1995.

[2] Achour, C. Ben, “Guiding Scenario Authoring”, In *Proceedings of the 8th European Japanese Conference on Information Modelling and Knowledge Bases,* Ellivuori, Finland, 1998, pp. 181-200.

[3] Ambriola V. and Gervasi, V., “Processing Natural Language Requirements”, In *Proc. of the 12th International Conference on Automated Software Engineering,* Los Alamitos, IEEE Computer Society Press, 1997, pp 36-45.

[4] Berry, D., Kamsties, E. and Krieger, M. “*From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity. A Handbook*”, Technical Report, University of Waterloo, Ontario, Canada, 2003, http://se.uwaterloo.ca/~dberry/handbook/ambiguityHandbook.pdf

[5] Berry, D. and Kamsties, E., “The Syntactically Dangerous *All* and Plural in Specifications” In *IEEE Software*, The IEEE Computer Society, Jan/Feb 2005.

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

[7] Calvo, H., Gelbukh, A. and Kilgarriff, A., “Distributional Thesaurus vs. WordNet: A Comparison of Backoff Techniques for Unsupervised PP Attachment”, In *Proceedings of Sixth International Conference on Intelligent Text Processing and Computational Linguistics: CICLING05,* Mexico City, Mexico, 2005, pp. 172-182.

[8] Daelemans, W., Zavrel, J., Van der Sloot, K. and Van den Bosch, A., *TiMBL: Tilburg Memory Based Learner, version 5.0, Reference Guide,* ILK Technical Report Series 03-10, 2003.

[9] d'Avila Garcez, A,, Russo, A., Nuseibeh, B. and Kramer, J., “Combining Abductive Reasoning and Inductive Learning to Evolve Requirements Specifications” In *IEE Proceedings – Software, 150(1)*, 2003, pp. 25-38.

[10] Fuchs N. and Schwitter, R., “Attempto Controlled English (ACE)”, in *Proceedings of the First International Workshop on Controlled Language Applications*, Katholieke Universiteit Leuven, Belgium, 1996.

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

[12] Gervasi V. and Nuseibeh B. “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] Gillon, B. S. “Ambiguity, Generality and Indeterminacy: Test and Definitions”. In *Synthese* 85:391-416, Kluwer Academic Publishers, The Netherlands, 1990.

[14] Gillon, B. S. “Ambiguity, Indeterminacy, Deixis and Vagueness: Evidence and Theory”, In *Semantics: A Reader* Oxford University Press, S. Davis and B. Gillon (eds.), 2003, pp. 157-187.

[15] Goldin, L. and Berry, D. “AbstFinder, A Prototype Natural Language Text Abstraction Finder for Use in Requirements Elicitation”, In *Automated Software Engineering,* volume 4, number 4, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1997, pp 375-412.

[16] Kamsties, E., "Surfacing Ambiguity in Natural Language Requirements", *PhD Thesis*, Fraunhofer-Institue für Experimentelles Software Engineering, Kaiserslautern, Germany, 2001.

[17] Kamsties, E., Berry, D.M., and Paech, B., "Detecting Ambiguities in Requirements Documents Using Inspections", in *Proceedings of the First Workshop on Inspection in Software Engineering (WISE'01)*, eds. M. Lawford and D. L. Parnas, Paris, France, 2001, pp. 68-80.

[18] Kilgarriff, A., “Thesauruses for Natural Language Processing”, in *Proceedings of NLP-KE*, Beijing, China 2003.

[19] Kilgarriff, A., Rychly, P., Smrz, P. and Tugwell, D., “The Sketch Engine”, In *Proceedings of Euralex 2004,* Lorient, France, 2004, pp. 105-116.

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

[21] Lin, D. “Automatic retrieval and clustering of similar words”. In *Proceedings of COLING-ACL '98*, Montreal, Canada, 1998, pp. 768-774.

[22] McLauchlan, M., “Thesauruses for Prepositional Phrase Attachment”, In *Proceedings of CoNLL-2004,* eds. Hwee Tou Ng and Ellen Riloff, Boston, MA, USA, 2004, pp. 73-80.

[23] Natt och Dag, J., Regnell, B., Gervasi, V. and Brinkkemper, S., “A Linguistic-Engineering Approach to Large-Scale Requirements Management”, In *IEEE Software*, The IEEE Computer Society, Jan/Feb 2005.

[24] Nenadic, G., Spasic, I. and Ananiadou, S., “Mining Biomedical Abstracts: What is in a Term?”, In *Proceedings of IJCNLP (International Joint Conference on Natural Language Processing),* Sanya City, Hainan Island, China, 2004.

[25] Porter, A.A., Votta L. G. and Basili, V. R., “Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment”, In *IEEE Transactions on Software Engineering,* Volume 21, Number 6, June 1995, pp. 563-575.

[26] Rada, R., Mili, H., Bicknell, E. and Blettner, M. “Development and application of a metric on semantic nets”, In *IEEE Transactions on Systems, Man, and Cybernetics,* 19(1), 1989, pp. 17-30.

[27] Resnik, P., “Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language”, In *Journal of Artificial Intelligence Research*, volume 11, 1999, pp. 95-130.

[28] Reubenstein H.B. and Waters, R. C., “The requirements apprentice: An initial scenario”, In S. Greenspan, editor, *Proc. of the 5th International Workshop on Software Specification and Design*, IEEE Computer Society Press, Pittsburgh, PA, May 1989, pp. 211-218.

[29] Rupp, C., “Linguistic methods of Requirements Engineering (NLP)”, In *Proceedings of European Software Process Improvement (EuroSPI 2000)*, Copenhagen, Denmark, 2000.

[30] Ryser, J. and Glinz, M., *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.

[31] Shull, F. Rus, I. and Basili, V., “How Perspective-Based Reading Can Improve Requirements Inspections”, In *IEEE Computer*, Vol. 33, No. 7, July 2000.

[32] Spanoudakis G., d’Avila Garcez A., and Zisman A., “Revising Rules to Capture Requirements Traceability Relations: A Machine Learning Approach”, In *Proceedings of 15th International Conference in Software Engineering and Knowledge Engineering (SEKE 2003),* San Francisco, CA, USA, 2003.

[33] Wasow, T., Perfors, A. and Beaver, D., “The Puzzle of Ambiguity” In O. Orgun and P. Sells (eds.) *Morphology and the Web of Grammar: Essays in Memory of Steven G. Lapointe*, CSLI Publications, 2003.

[34] Weiss, S. and Kulikowski, C.A., *Computer systems that learn: classification and prediction methods from statistics, neural nets, machine learning, and expert systems*, Morgan Kaufmann Publishers, 1991.

[35] West M.M. and McCluskey T. L., “The Application of a Machine Learning Tool to the Validation of an Air Traffic Control Domain Theory”, In *Proceedings of 12th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)”*, Vancouver, Canada, 2000.

[36] Zowghi, D., Gervasi, V. and McRae, A., “Using default reasoning to discover inconsistencies in natural language requirements”, In *Proc. of the 8th Asia-Pacific Software Engineering Conference*, 2001, Macau, China, pp. 133-140.

Appendix

Sentences from our Corpus used in Ambiguity Surveys	AA	UA	DS	SS	NN	PLD	ML
(It) manages ...... coordinate [[systems] and other related objects]	N	N	N	Y	s	l	I
(It) must be configured with the proper [[item] and system components]	N	N	Y	Y	m	l	I
(It) cannot function with the proper [[installation] and configuration]	N	N	Y	N	s	z	I
(It) shall display categorized [[instructions] and documentation]	Y	Y	N	Y	m	z	N
The original [[meeting date] or location] may then need to be changed	N	N	Y	Y	s	h	I
(It is) useful in determining a best [[meeting date] and location]	N	N	Y	Y	s	h	I
Best [[meeting dates] and locations] should be determined	N	N	Y	Y	s	h	I
admission information, medication, equipments, daily [[record] and history]	N	N	Y	Y	s	z	I
build vehicles from mass-produced [[parts] and subassemblies]	N	Y	N	N	s	z	N
(They're) assembled into ...... vehicle models for model-based [[design] and development]	Y	Y	Y	Y	s	z	N
It also lists applicable [[design constraints] and system attributes]	Y	Y	N	N	s	z	N
Create (them) from the architectural [[description] and component characterizations]	N	N	N	Y	m	l	I
There are several [[assumptions] and dependencies]	N	N	N	N	s	z	I
The system will parse ...... scalar [[input] and output ports]	N	Y	Y	Y	s	l	N
(It is) describing the size of vector-based [[inputs] and outputs]	N	N	Y	Y	s	z	I
For vector-based [[input] and output], the system will provide ...... characterization	N	N	Y	Y	s	z	I
The user may define architectural components and connectors	N	Y	N	Y	s	z	N
Non-Functional User's [[guide] and help documentation]	Y	Y	N	N	s	l	N
Signal [[units] and data transfer protocol] must match	N	N	N	Y	m	l	I
Revamp the current ...... [[hardware] and software]	N	N	Y	N	s	z	I
Take care of business [[rules] and transactions]	Y	Y	N	Y	s	z	N
Involving ...... [election and/or [geopolitics]] entities	Y	Y	N	N	s	z	N
Keep track of any [modification or [access]] to the database	N	N	Y	Y	s	z	I
(It) might be automatically [[rejected] or flagged]	N	Y	N	Y	n	z	N
Initialization [[rules] or pre-conditions]	N	N	N	Y	s	z	I
Termination [[rules] or post-conditions]	N	N	N	N	s	z	I
Activity like ...... [verification or [printing]] of voters' lists	N	Y	N	Y	n	z	N
(It might be) [rejected or [flagged]] for further processing	N	N	N	Y	n	z	I
Display categorized [[instructions] and documentation]	Y	Y	N	Y	m	z	N
Minimize the [time span and [human resources]] of regular inspection	N	Y	N	Y	m	z	N
Facilitate the [scheduling and [performing]] of works	N	N	N	Y	n	z	I
Connectivity of all [control flow and [data flow signals]] between sub-models	N	Y	N	Y	m	h	N
This should lead to reduced [[cycle times] and costs] for automotive manufactures	N	N	N	Y	s	h	I
Patients conditions are [inspected and [recorded]] automatically	Y	Y	N	N	n	z	N
[Constraints ...... and [dependencies]] that apply to the product	N	N	N	Y	s	z	I
[Assumptions and [dependencies]] that are of importance	N	N	N	N	s	z	I
It is ...... very [[common] and ubiquitous]	N	N	N	Y	n	z	I
(It) targeted the [project and [election]] managers	N	N	N	N	s	z	I
Contributing to the [recording and [accuracy]] of the data	N	N	N	Y	s	z	I
Proceed to [enter and [verify]] the data	N	N	N	N	n	z	I
(He) has the ability to [generate and [print]] pre-defined reports	N	N	N	Y	n	z	I
[Reliability and [security]] considerations	Y	N	N	N	s	z	N
Risk greatly increased by the lack of [[funding] and local resources]	N	N	Y	Y	m	l	I
[Operating ...... and [performance]] requirements	N	Y	Y	N	s	z	N
Taken with respect to the [quantity and [type]] of data	N	N	Y	Y	s	z	I
Used for [verification and [clean-up]] purposes	Y	Y	N	N	s	z	N
(It) will be [implemented and [executed]] on the ...... platform	N	N	Y	Y	n	z	I
[Memory or [hard disk]] space	N	Y	Y	Y	s	l	N
[Requirements or [data]] storing	N	Y	N	N	m	z	N
[Communication and [performance]] requirements	Y	N	N	Y	s	z	N
[Assumptions and [dependencies]] that apply to the product	N	N	N	N	s	z	I
[Vandalism or [act]] of God	N	N	N	Y	s	z	I

Key: AA = Acknowledged Ambiguity, UA = Unacknowledged Ambiguity, NN = Noun Number (s=same, m=mixed, n=n/a), PLD = Phrase Length Difference (l = lower than 0, z = 0, h = higher than 0), DS = Distributional Similarity, SS = Semantic Similarity, ML = Judgement of ML Algorithm (N = nocuous, I = innocuous).

Note: The external modifier is underlined.
The outer set of square brackets indicate coordination first; the inner set indicate coordination last.
"……" indicates elided words that are not necessary for understanding the sentence.
"( )" indicates replacement of something that is unnecessarily complex by something generic.