Analysing Anaphoric Ambiguity in Natural Language Requirements

Hui Yang • Anne de Roeck • Vincenzo Gervasi • Alistair Willis • Bashar Nuseibeh

Abstract: Many requirements documents are written in natural language (NL). However, with the flexibility of NL comes the risk of introducing unwanted ambiguities in the requirements and misunderstandings between stakeholders. In this paper, we describe an automated approach to identify potentially nocuous ambiguity, which occurs when text is interpreted differently by different readers. We concentrate on anaphoric ambiguity, which occurs when readers may disagree on how pronouns should be interpreted.

We describe a number of heuristics, each of which captures information that may lead a reader to favour a particular interpretation of the text. We use these heuristics to build a classifier, which in turn predicts the degree to which particular interpretations are preferred. We collected multiple human judgements on the interpretation of requirements exhibiting anaphoric ambiguity, and show how the distribution of these judgements can be used to assess whether a particular instance of ambiguity is nocuous. Given a requirements document written in natural language, our approach can identify sentences which contain anaphoric ambiguity, and use the classifier to alert the requirements writer of text that runs the risk of misinterpretation.

We report on a series of experiments that we conducted to evaluate the performance of the automated system we developed to support our approach. The results show that the system achieves high recall with a consistent improvement on baseline precision subject to some ambiguity tolerance levels, allowing us to explore and highlight realistic and potentially problematic ambiguities in actual requirements documents.

Keywords: *nocuous ambiguity; natural language requirements; anaphoric ambiguity; noun-phrase coreference resolution; antecedent preference heuristics; human judgements; machine learning*

1. Introduction

In industrial practice, the vast majority of requirements documents are written in natural language (NL) [3], and so run the risk of being ambiguous. Ambiguity is a phenomenon inherent in natural language, and occurs when a linguistic expression may be understood in two or more different ways. Ambiguity has often been considered a potentially harmful attribute of requirements [6], since ambiguity in the requirements can potentially lead to specifications which do not accurately describe the desired behaviour of the system to be developed. For example, if the customer's interpretation of the requirements is not the same as that of the system's developers and testers, then the system might not be accepted after customer validation. Similarly, if a user manual is written based on a different interpretation of the requirements from that of the developers, the system runs the risk of being erroneously documented.

Berry and his colleagues [3] illustrated the ambiguity phenomenon in requirements documents, and, following common practice in linguistics, classified them into four main categories, depending on whether the source of the ambiguity lies at the level of words (lexical ambiguity), syntax (syntactic ambiguity), semantic interpretation (semantic ambiguity), or the interaction between interpretation and context (pragmatic ambiguity). Previous work on ambiguity in RE mainly focused on the first three ambiguity types, and attempted to address the problem from at least two perspectives.

- Providing users with a restricted NL [6, 13], tool [19], or handbook [1, 3, 4, 30] to assist with writing less ambiguously.
- Detecting ambiguity by examining the text using lexical, syntactic, or semantic information, or with the help of quality metrics [23, 24, 28, 32, 34].

However, these attempts at solving the problems introduced by the use of natural language in requirements run the risk of failing to retain the benefits. These include the universality of natural language as a medium of communication, and the fact that the very same ambiguity can be usefully exploited to convey multiple acceptable meanings, to leave leeway for future negotiations, or to avoid over-constraining the problem at too early a point in the development cycle [17].

Our approach to ambiguity analysis in NL requirements therefore differs from earlier work. We aim to help requirements analysts to focus on identifying only those instances of ambiguity that are *likely* to lead to misunderstandings between stakeholders (because different stakeholders select different interpretations), while discounting those that are unlikely to cause misunderstandings (because all stakeholders choose the same interpretation). The analyst can then perform a more careful analysis of the situation (an analysis that is better left to human judgement), and decide whether more elicitation is needed, or whether the form in which the requirement is expressed needs to be made more precise – or possibly, formalized[1]. For example:

E1. *Another feature of GRASS is **its** ability to use raster, or cell, data.*

[1] As with programming, requirements are inherently an ill-conditioned problem; small changes in their interpretation can lead to hugely differently behaviour of the developed systems. We do not address here the issue of estimating the *effect* of different interpretations caused by ambiguity.

E2. *Table data is dumped into a delimited text file, which is sent to the remote site where* ***it*** *is loaded into the destination database.*

Note that both of these examples are ambiguous, because there are multiple ways of choosing a referent for the pronoun. In our studies of these examples described later in the paper, 12 out of 13 readers interpreted the pronoun '*its*' in example (E1) as referring to '*GRASS*'. In practice, this means that this text is unlikely to lead to misunderstandings between stakeholders and the ambiguity is therefore innocuous. However, almost half of readers interpreted the pronoun 'it' in example (E2) as '*Table data*', while the other half interpreted it as '*a delimited text file*'. This suggests a higher risk that different readers will interpret the text in different ways, and so lead to miscommunication in the development process. The ambiguity is therefore *nocuous*. It implies that not all cases of anaphoric ambiguity are potentially harmful.

Ambiguity is therefore not a property just of a text, but a conjoint property of the text and of the interpretations held by a group of readers of that text [9, 54]. As a consequence, any ambiguity presented in a requirement can be innocuous in a certain context (for example, if domain knowledge shared between customers, analysts and programmers lead them all to prefer the same interpretation) and nocuous in others (for example, when the same text is forwarded to a translator who is in charge of translating the user manual to a different language, and who has little knowledge of the particular domain). The objective of our research is thus to develop an automated technique that can classify ambiguities in NL requirements as nocuous or innocuous for a general audience, to vary the sensitivity of the analysis depending on the readership, and to inform the analyst of the potentially dangerous cases.

To bring all potentially ambiguous text to the attention of the analyst would not be very efficient, as innocuous cases do not carry a high risk of misunderstanding. In contrast, bringing only the likely nocuous cases to the attention of the analyst is much more valuable, as it allows the analyst to concentrate on rephrasing those examples which run the risk of being misunderstood at a later stage in the development cycle. The work in this paper builds on Yang et al. [56] which investigates the linguistic factors that contribute to the preference for particular readings, and demonstrates that the distinction between nocuous and innocuous ambiguity can be characterized precisely, and can be implemented in a computational model. The approach builds on previous work applied to requirements documents [9, 54, 57], which focused on coordination ambiguity shown in example E3 below:

E3. *They support a typing system for architectural components and connectors.*

In the example (E3), the coordination construction '*architectural components and connectors*' can be bracketed as '*[architectural [components and connectors]]*' or as '*[architectural components] and [connectors]*. We developed a set of heuristics to automatically predict whether a coordination ambiguity may be misinterpreted given an *ambiguity threshold* (which indicates the degree of nocuous ambiguity that can be tolerated for a given domain of application). For example, we may wish to tolerate less ambiguity in high risk domains, where failure carries a higher cost, and insist that all requirements be formulated in utterly precise (even formal) terms, at the expense of ease of expression. In contrast, in lower risk domains we might be prepared to accept more ambiguity as

long as that encourages creativity in inventing requirements, for example when adopting a rapid-prototyping process so that any misunderstanding can be corrected early anyway.

In this paper, we investigate anaphoric ambiguity exemplified by use of pronouns such as '*it*', '*them*' and '*their*'. These are common in requirements documents and have been noted as being problematic to analyse because contextual dependencies have to be taken into account [3]. There may be several items in the context to which a pronoun might refer, and those items might be spread over several previous sentences.

Our goal is therefore to develop an architecture of an automated system to support requirements writing, by incorporating nocuity detection into the requirements workflow. At the centre of such an architecture is a classifier which can determine automatically whether an instance of anaphoric ambiguity is nocuous or innocuous. We develop the classifier using instances of anaphoric ambiguity extracted from a collection of requirements documents. For each instance, a set of human judgements are used to classify the ambiguity as nocuous or innocuous. A classifier is then trained on the linguistic features of the text and the distribution of judgements to identify instances of nocuous ambiguity in new cases. The classifier is then integrated with functional modules to extract ambiguous anaphora instances from full-text documents and use the context to identify those instances which display nocuous ambiguity.

In the work described in this paper, we followed the general methodology for automatic identification of nocuous ambiguity presented in Yang et al. [56]. We implement, refine and extend the preliminary model of ambiguity described in Yang et al. [58] by presenting an overall system architecture which consists of four major functional modules. This includes two new functional modules: Text Preprocessing and Ambiguous Instance Detection, which include a noun-phrase (NP) coreference resolution engine. Moreover, we explore more antecedent preference heuristics to further refine and improve the antecedent classifier. We conduct a new set of experiments, and report our experimental results to illustrate and evaluate the extended ambiguity model.

The remainder of the paper is organized as follows. Section 2 provides some background on anaphoric ambiguity, with relevant anaphora examples. Section 3 describes an overall system architecture for automatic detection of nocuous ambiguities. In Section 4 we give details about the identification of anaphoric ambiguity instances and the extraction of associated antecedent candidate with the help of a noun-phrase coreference resolution engine. Section 5 discusses the building of antecedent classifier, which is underpinned by three main components, antecedent preference heuristics, human judgments and machine learning. In Section 6, we introduce the procedure for automatic judgment of nocuous ambiguities. Our experimental setup and results are reported in Section 7, and Section 8 addresses potential threats to validity. Section 9 reflects on related work, and is followed by our conclusions and plans for future work.

2. Anaphoric ambiguity

2.1 Nocuous Anaphoric Ambiguity

An *anaphor*[2] is a linguistic expression which refers to a preceding utterance in text. Similarly, a *cataphor* refers to an expression which occurs further on in the text. For example[3],

E4: *A prototype exists and **it** will be examined for reuse.* ('*it*' is an anaphor which refers to '*a prototype*')
E5: *If **they** are valid, these parameters will be stored in the data processing subsystem.* ('*they*' is a cataphor which refers to '*these parameters*')

Compared with anaphora, cataphoras are relatively uncommon in natural language documents, and we have found very few instances in the requirements documents that we are working with. In this paper, we are concerned with anaphora only. Specifically, we focus on anaphoric references through three common-used pronoun types in requirements documents: 3rd personal pronouns (e.g., *it, they, them*), possessive pronouns (e.g., *their*, *its*) and prepositional pronouns (e.g., *under it*). The reason to concentrate on these three kinds of pronouns is that: (a) we found them to be widespread in requirements documents; for example, in our collected dataset that consisted of 11 requirements documents with a total of 26, 829 sentences, 1642 (about 6.12%) sentences contained at least one 3rd personal or possessive pronouns; (b) Due to the high cost in human judgment collection, we were limited in the number of instances on which we could collect human judgments. To make sure that each pronoun type has enough cases for machine learning, we decide to first focus on commonly used personal and possessive pronouns, and then extend our research to other kinds of pronouns, such as demonstrative pronouns (e.g., *this*, *these*), indefinite pronouns (e.g., *each*, *some*), in the future work. We selected a set of 200 anaphoric instances for the research described in this paper.

The expression to which an anaphor refers is called its *antecedent*. Antecedents for personal pronoun anaphora are nouns or noun phrases (NPs) found elsewhere in the text, usually preceding the anaphor itself. In (E4), the pronoun '*it*' is an anaphor and *'A prototype'* is the antecedent NP. The interpretation of (E4) states that the prototype will be examined for reuse. The intuition is that the anaphor is a placeholder for the antecedent that has been mentioned previously.

Anaphoric ambiguity occurs when the text offers two or more potential antecedent candidates either in the same sentence or in a preceding one, as in example (E6).

E6. *The procedure shall convert the 24 bit image to an 8 bit image, then display **it** in a dynamic window.*

In this case, either of the three underlined NPs could be legitimate antecedents for the anaphor: a procedure that displays the 24 bit image would meet the requirement, as would one that displays

---

[2] Plural: *anaphora*
[3] Our examples are adapted (in many cases abbreviated) from our collection of requirements documents. We render anaphora in bold and underline antecedent candidates.

the 8 bit image. It is possible for different stakeholders to commit, legitimately, to different, incompatible readings of the same requirement. The requirement can be signed off by all stakeholders, and yet the developer could produce a system displaying, for example, the 8 bit image - which would not satisfy the customer's intention of having the 24 bit image displayed instead. The final result of this sort of misunderstanding is then the delivery of a system that does not satisfy the customer.

Now consider examples (E7) and (E8), both of which have more than one antecedent candidate for the anaphor *'it'*.

E7. *The MANUAL schema models the system behavior when **it** is in manual override mode.*
E8. *A prototype exists for this process and **it** will be examined for reuse.*

We asked 13 people with software engineering backgrounds to identify which NP they thought the anaphor referred to in (E7). 7 committed to "*the MANUAL schema*", whereas 6 chose '*the system behavior*'. Hence, we consider the instance (E7) as a typical *nocuous* ambiguity case for study. In contrast, for (E8), all judges agreed that '*a prototype*' is the antecedent. This illustrates the difference between nocuous and innocuous ambiguity. Both (E7) and (E8) contained more than one possible antecedent, but in the case of (E7) the interpretations of stakeholders significantly diverged, whereas all stakeholders committed to the same antecedent in (E8). We identify (E7) as a case of *nocuous anaphoric ambiguity* and (E8) as a case of *innocuous anaphoric ambiguity*.

These examples illustrate three further points:

1. Merely identifying the presence of all potential anaphoric ambiguity in requirements documents (as suggested, among others, by Gnesi et al. [18], where this feature is called '*implicity*') is not an effective approach, due to the high number of false positives. It is almost always the case that there exist multiple possible antecedents for the anaphor, but readers will often agree on the same interpretation of the text (as illustrated in example (E8)). The point is to identify those cases where readers may disagree.
2. The traditional computational approach to ambiguity, of trying to automatically determining the correct antecedent ('*disambiguation*') is inappropriate since stakeholders would still hold incompatible interpretations.
3. The problem with example (E7) is that different users may interpret the text differently; finding a most likely interpretation would be of no help as part of the development process. No individual stakeholder can be aware of whether other stakeholders hold different interpretations from his or her own. Therefore, our approach is to alert users to where nocuous ambiguities might occur.

2.2 Human Judgments

We define an ambiguity as nocuous if it gives rise to diverging interpretations. We concur with Wasow et al. [53] who suggests that ambiguity is always a product of multiple denotations to a linguistic expression that the interpreter assigns to, and thus is a subjective phenomenon. From this perspective, modeling interpretations requires access to human judgments, which we capture by

surveying participants. Given a linguistic expression, we ask human judges for their interpretations of it. We use this information to decide whether, given some ambiguity threshold, a particular instance is to be seen as innocuous or nocuous depending on the degree of dissent between the judges.

2.2.1 The Building of the Dataset

Anaphora instances in requirements documents. We collected a set of 11 requirements documents from RE@UTS web pages[4]. The documents specify systems from a variety of application domains, including transportation, engineering, communication, and web applications. In this dataset, we manually located a set of 200 anaphora instances [5]containing different types of pronouns. Each instance consists of one or two sentences - i.e. the current sentence in which the pronoun appears, and the preceding one, depending on the position of the pronoun in the sentence. Each instance has two or more possible noun phrase (NP) candidates for the antecedent of an anaphor. In our data, nearly half of the cases (48%) were subject pronouns (i.e. '*it*', '*they*') although objective (i.e. '*them*', '*it*') and possessive (i.e. '*its*', '*their*') pronouns also accounted for a significant number (15% and 33%, respectively). Prepositional pronouns (e.g., '*under it*', '*on them*') were relatively rare (4% only - 8 instances in the whole dataset).

Human judgment collection. The anaphora instances containing potential ambiguity were randomly partitioned into five separate surveys (each survey containing 40 instances), which were then administered to a group of 38 respondents (computing professionals who were academic staff, research students, or software developers) through a web site[6]. Among these judges, 25 were native English speakers, 22 people had at least 3 years RE/SE experience and 11 for 1~ 3 years, and 10 had some background in natural language processing (NLP). Each session started with an introduction explaining the task and illustrating how to record a judgment. The judges were then asked to record their interpretations for each instance in the survey. Instances were presented by highlighting the anaphor and each candidate antecedent (as in the example in Table 1 discussed later). The judges were asked to select the antecedent candidate they thought corresponded to the anaphor. To avoid bias and to collect opinions from different judges, each judge was allowed to complete a survey only once (but could complete several distinct surveys). In our collection, to ensure enough judges and to minimize the effect of noise introduced by rogue judgments[7] [25], each instance was judged by at least 13 people, and tolerate up to one or two rogue judges. Each judge was allowed to choose one of NP candidates. If the judge could not decide which one is preferred, the alternative option '*I am not sure which one of the above is more appropriate*' was available for selection.

---

[4] http://research.it.uts.edu.au/re/

[5] In requirements documents, the sentences that describe requirements are not generally specified by the writer. So anaphora instances are collected based on the whole text of the document other than some specific requirements sentences.

[6] http://www.surveymonkey.com/s.aspx?sm=kbGtRdJJXqWabZFk28tJfw_3d_3d

[7] Rogue judgments are errors made by judges through carelessness or by accident, rather than judgments that reflect a genuine difference of opinion.

2.2.2 Ambiguity Threshold

We use the *ambiguity threshold* as a key concept in determining when an ambiguity is nocuous. Different application areas may be more or less tolerant of ambiguity [43]. For instance, requirements documents describing safety critical systems should seek to avoid misunderstandings between stakeholders. Other cases, such as music books, could be less sensitive.

In order to predict whether a particular instance displays nocuous ambiguity, we used the concept of ambiguity threshold proposed by Willis et al. [54], which sought to implement a flexible tolerance level to nocuous ambiguity. We adapted the definition specifically to anaphoric ambiguity. The definition makes use of a notion of *certainty* of a particular NP antecedent candidate, which is calculated as the percentage of the judgments that favoured this NP against the total judgments in the ambiguity instance. For instance, consider the example in Table 1. The certainty of the NP '*supervisors*' is 12/13=92.3% and the certainty of the NP '*tasks*' is 1/13=7.7%. The definition of ambiguity threshold, adapted to anaphoric ambiguity is as follows:

Definition. Given an anaphora instance, *I*, a collection of judgments associated with NP antecedent candidates of the anaphor, and an ambiguity threshold τ (where $0.5 < \tau \leq 1.0$):

If there is one NP candidate that has a *certainty* greater than τ, then the instance *I* exhibits *innocuous* ambiguity at the threshold τ. Otherwise, the instance *I* exhibits *nocuous* ambiguity at the threshold τ.

In this definition, the ambiguity threshold τ is set to be greater than 0.5. This constraint guarantees that, in an *innocuous* ambiguity case, only one NP candidate can be a potential anaphora reference. , To illustrate the definition, at the ambiguity threshold $\tau = 0.8$, the ambiguity in Table 1 is *innocuous* because the certainty of the NP *'supervisors'* exceeds the threshold, reflecting a clear agreement between the judges.

Table 1. Judgment count for an anaphoric ambiguity instance

1. Supervisors may only modify tasks **they** supervise to the agents they supervise.		
	Response Percent	Response Count
(a) supervisors	92.3%	12
(b) tasks	7.7%	1

Figure 1 depicts the relationship between the ambiguity threshold and the incidence of nocuous ambiguity. As expected, the number of nocuous ambiguities increases with the ambiguity threshold τ. This is understandable because high thresholds mean low tolerance to the disagreement among the judges, which is usually hard to achieve. For high thresholds (e.g., $\tau \geq 0.9$), more than 60% of anaphoric ambiguity instances are classified as nocuous, showing that consensus between judges on an interpretation is elusive for context-dependent cases like anaphora. In practice, however, judges will make some mistakes and it is advisable to allow some degree of ambiguity tolerance (e.g., $0.7 \leq \tau \leq 0.9$) before registering a genuine difference of opinion indicating that an ambiguity is nocuous.

Ambiguities that are nocuous at a lower threshold are much harder to detect than those that are nocuous only at higher thresholds because, by definition, the degree of divergence between

judgments will be more obscure at lower thresholds. Thus, τ can be used to find the most nocuous instances by comparing the nocuous ambiguity lists at different levels of thresholds, allowing an analyst to focus on the most critical cases first (e.g., when limited time is available for further elicitation).

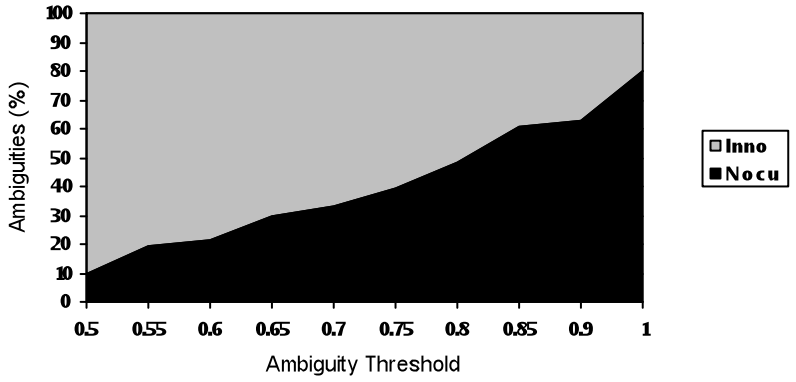

Fig. 1 Proportions of interpretations at different ambiguity thresholds in the anaphora instances

3. Overall System Architecture

We developed an automated system to detect nocuous anaphoric ambiguities from full-text documents. The system architecture is shown in Figure 2. The initial input is a complete requirements document. The output is the set of selected sentences that contain potentially nocuous anaphoric ambiguities.

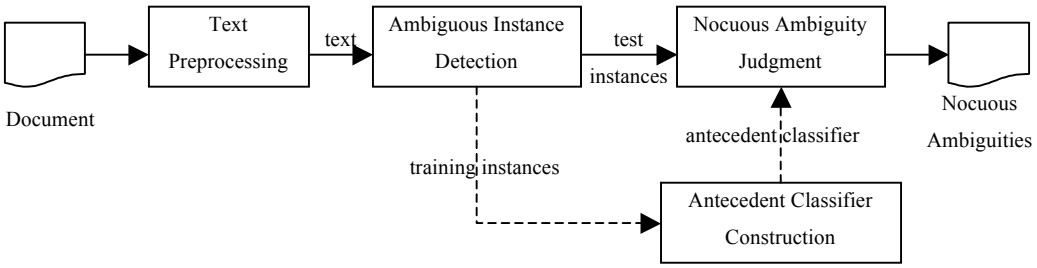

Fig. 2 Overall system architecture

The system consists of four major functional process modules.

(a) **Text Preprocessing Module.** The input requirements document is split into separate sentences using an established sentence boundary detector[8]. The individual sentences are then passed to the Genia Tagger[9] [50] which identifies the individual words' part of speech, and marks phrase boundaries. For example, the Genia tagger marks the phrase boundaries in (E6) above as:

[A prototype]$_{NP}$[exists]$_{VP}$[for]$_{PP}$[this process]$_{NP}$ and [it]$_{NP}$[will be examined]$_{VP}$[for]$_{PP}$[reuse]$_{NP}$

[8] http://text0.mib.man.ac.uk:8080/scottpiao/sent_dectector

Notice particularly that the two NPs '*a prototype*' and '*this process*' are identified, as is the pronoun '*it*' which is also highlighted as an NP. Any additional statistical information (e.g., word co-occurrence) that is required for the heuristics in the classifier module are also saved into the back-end database at this point.

(b) **Ambiguous Instance Detection Module.** This module identifies a set of possible antecedents for each identified pronoun. From the output of the text pre-processing module, three types of pronouns, i.e. 3rd personal, possessive, and prepositional pronouns (e.g., *it*, *them*, *their, under it*) can be identified, as can the set of all preceding NPs. In fact, it is possible for several NPs to refer to the same entity. The NPs are therefore clustered into corefering groups; this process is discussed fully in section 4.2. Clearly, in a whole document there will usually be too many NPs preceding the pronoun for them all to be possible antecedents. In practice, we take the set of possible antecedents to be the NPs preceding the pronoun in the same sentence, and (if there is only one preceding NP in the sentence), the sentence before it.

(c) **Antecedent Classifier Construction Module.** This module implements a classifier based on multiple human judgments of the most likely NP antecedent candidate in terms of an anaphoric ambiguity instance. A number of preference heuristics are also included to model the factors that may favour a particular interpretation. The trained antecedent classifier feeds into the nocuous ambiguity judgement module. More detained discussions will be given in the later section 5.

(d) **Nocuous Ambiguity Judgment Module.** Finally, for each pronoun and associated NP antecedent candidates, the antecedent classifier assigns one of the antecedent preference labels, *Positive* (Y), *Questionable* (Q), and *Negative* (N), to each candidate. This information is then used to calculate whether the anaphoric ambiguity is nocuous or innocuous, with the nocuous ambiguities being presented to the user as the final step.

In the following sections, we will separately discuss the behaviour of the three modules, Ambiguous Instance Detection, Antecedent Classifier Construction, and Nocuous Ambiguity Judgment.

4. Ambiguous Instance Detection

To identify nocuous anaphoric ambiguities, each anaphor must be identified, along with its potential antecedents. The particular instance of the anaphor and the candidate antecedents (clustered into corefering sets [46]) are then used either as training instances to train the antecedent classifier, or as input to the judgement module during live deployment (Figure 3).

[9] http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/tagger/

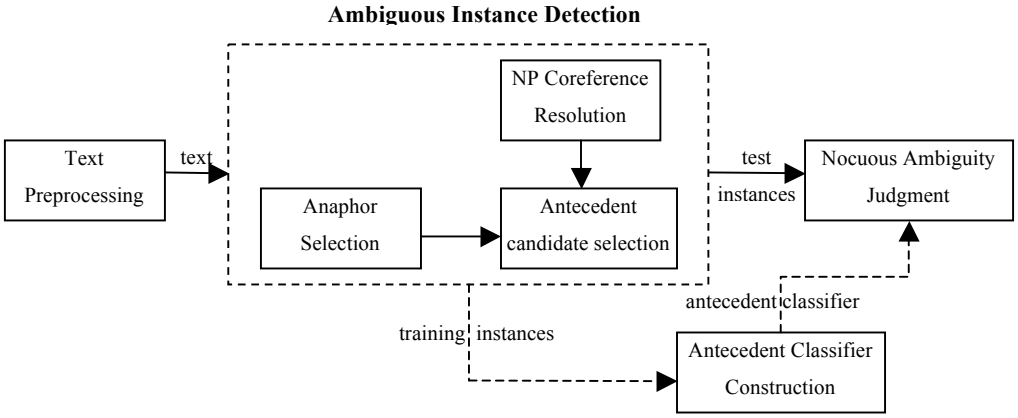

Fig. 3 The functional architecture of the ambiguity instance detection module (highlighted by the dashed line)

4.1 Selection of Anaphora and Candidate Antecedents

As described earlier, the output of the text preprocessing module is the set of separated sentences that make up the requirements document. For each sentence, word token information (e.g., word, word lemma, POS tag) and phrase chunks are identified by the Genia Tagger. The phrases (specifically the NPs) provide the necessary antecedent cases. Selection of anaphora and possible antecedent candidates proceeds by:

A. The occurrence of the concerned pronouns is located by exact word token matching. Instances of anaphora are taken to be the 3rd personal and possessive pronouns, i.e. *it*, *they*, *them*, *its* and *their*. The rationale for limiting ourselves to the 3rd person is that, in requirements documents, very rarely are 1st and 2nd person references (e.g., '*you*' or '*my'*) used at all. This assumption does not hold for other kind of documents that are of relevance to requirements engineering, e.g. in contracts they appear commonly (as in, '*our service*' and '*your obligations*'). Our approach could of course be extended to address these.

B. The candidate antecedents are the two or more NPs found in the text preceding the pronoun.

C. If there are two or more NPs preceding the pronoun in the same sentence, then we use those as the candidate NPs. Otherwise, we also include the NPs in the previous adjacent sentence.

To illustrate point *C*, consider example (E7).

E9. *In most Consortium states, these users must go to a service center to view LMI data using the LMI ACCESS application.* ***They*** *will primarily use the LMI ACCESS Job Seeker module.*

In this case, there are no NPs (and so no candidate antecedents) preceding the pronoun '*They*'. Therefore, the set of candidate NPs is extended to those appearing in the previous sentence, that is, {*[most Consortium states]* $_{NP}$*, [these users]* $_{NP}$*, [a service center]* $_{NP}$*, [LMI data]* $_{NP}$*, [the LMI ACCESS application]* $_{NP}$}.

However, two types of NP compounds are given special consideration when identifying candidate antecedents:

(a) NPs with preposition (PP) attachments are extracted as distinct candidate antecedents. For example, if the Genia tagger identifies the NP and preposition phrases:

[job listings] $_{NP}$ *[from]* $_{PP}$ *[other job banks]* $_{NP}$

then the two distinct NPs, '*job listings from other job banks*' and '*other job banks*' would both be considered as candidate antecedents, because the 2nd NP is normally considered part of the PP which is part of the first NP. Currently, our system only deals with some simple PP attachment cases, i.e. the NP only attached with one PP. The ambiguity problem in NPs with multiple PP attachments (e.g., '*the system for relocation parcels from Appraisals*') will be addressed in future work.

(b) Coordinated NPs (i.e. NPs conjoined with '*and*' or '*or*') are also extracted separately. For instance, the tagged text:

[several counties] $_{NP}$ *and [metropolitan areas]* $_{NP}$

is separated into three NP candidates: '*several counties and metropolitan areas*', '*several counties*', and '*metropolitan areas*'.

4.2 Noun-Phrase Coreference Resolution

Coreference occurs when multiple expressions in a sentence or document refer to the same entity in the world. Consider the instance (E10)

E10. *The Raw Data Processing Subsystem first checks the CCSDS parameters for valid values and stores these parameters if* ***they*** *are valid.*

This is a typical coreference case because the two NPs '*the CCSDS parameters*' and '*these parameters*' probably refer to the same parameters. When this example was presented to human judges, both of these NPs were separately selected as possible references of the pronoun '*they*' by our judges. Without coreference resolution, the system would treat these two NPs as unrelated NPs, and mistakenly recognize this anaphoric ambiguity as nocuous because of the lack of agreement between the judges. However, in practice, we would expect this to be an example of *innocuous* ambiguity because the two NPs actually refer to the same object, and therefore the judges were actually in agreement. Therefore, the ability to link coreferring noun phrases within an ambiguous instance is critical to accurately identifying nocuous ambiguity.

We build a noun-phrase coreference resolution engine which adopts the standard learning-based framework employed in the work by Soon et al [46]. A set of references (i.e. NPs), along with the coreference relationships among them, are manually extracted from a set of 200 anaphoric ambiguity instances. Note that this set of anaphoric ambiguity instances (described later) is also used for nocuous ambiguity analysis.

During training, for each anaphor ambiguity instance, training instances are created by pairing all possible NP antecedent candidates. These pairings are represented as a set of feature vectors. The pairings that contain coreferring NP phrases form positive instances, while those that contain two non-coreferent phrases form negative instances. Once the training instances are created, a

classifier is built by a k-nearest neighbour learning algorithm (See Section 4.2.2 for details about the building of a coreference classifier).

To find the coreference relations among the possible NPs antecedent candidates in a new ambiguous instance, potential pairs of coreferring NPs are presented to the classifier, which determines whether the two NPs corefer or not.

In our system, we use a heuristics-based method to exploit the factors that influence coreference determination. The heuristics are incorporated in terms of feature vectors. In the following subsections, we will introduce more details about the building of a noun-phrase coreference engine.

4.2.1 Coreference Heuristics

We use the mention-pair model [39] to build a NP coreference engine to determine which sets of NPs corefer.

Definition: A mention-pair coreference instance consists of two NPs, i.e. NP_i and NP_j, where NP_i appears preceding NP_j in the text, and NP_i is the referent of NP_j.

where NP_i and NP_j are non-pronoun noun phrases. For example, in the instance of (E8), the pairing '*the CCSDS parameters*' and '*these parameters*' make up a coreference instance, and '*the CCSDS parameters*' is the referent of '*these parameters*'.

To build the coreference engine, this definition is used, along with the set of features described in Table 2 (discuss later).

Different types of information are used by the heuristics, which include the string contained in the phrase, grammatical and syntactic features, and semantic class information (i.e., the type of object referenced by the phrase). Here we present below a brief description of each type of heuristics.

String-matching Heuristics. String-matching heuristics aim to capture the string matching patterns between the two coreferring NPs. To facilitate matching, for each NP, we first extract the information of its headword[10] and modifiers. We perform four types of string-matching: (a) **Full-string matching**: two strings are stripped of their stopwords, such as articles (e.g., '*a*', '*an*', '*the*'), demonstrative pronouns (e.g., '*this*', '*that*', '*these*', '*those*'), then compared with each other. Therefore, the phrase '*a test*' matches the phrase '*the test*'. (b) **Headword matching**: The phrases that share the same headword are more likely to be coreferent than those that have different headwords. For instance, '*the CCSDS parameter*' and '*these parameters*' in the example (E8) (c) **Modifier matching**: A modifier (e.g., adjective or adverb) could change or limit the meaning of a noun. It is possible that two NPs with the same headword but different modifiers refer to distinct entities. For example, '*super users*' and '*ordinary users*'. (d) **Alias name**: One phrase is an alias of the other phrase, e.g., '*AKSAS*' is the abbreviation of '*Alaska State Accounting System*'. The alias

[10] A *headword* is the main word of a phrase, and the other words in that phrase modify it. For example, for the noun phrase '*the CCSDS parameter*', '*parameter*' is the headword, and '*CCSDS*' is a noun modifier for the headword.

list is either collected via the abbreviation table provided the document or extracted from the text using an abbreviation extraction algorithm[11] [40].

Grammatical Heuristics. This type of heuristics explores the grammatical properties of one or both NPs involved in an instance, which includes (a) **NP type**: Some specific NPs, such as the definite NP (e.g., '*the test*') and demonstrative NP (e.g., '*these parameters*'), are likely to be coreferent with any NP that precedes it. (b) **Proper name**[12]: If two NPs are both proper names, they prefer to be coreferent, such as '*the Raw Data Processing Subsystem*'. (c) Number agreement: Coreferent NPs generally agree in number.

Syntactic Heuristics. Syntactic heuristics are measured based on the grammatical role that the NPs play in the syntactic structure of the sentence. Three syntactic heuristics are developed: (a) **Prepositional Phrase (PP) Attachment**: If one NP is the PP attachment of the other NP, they have preference against coreference. For example, as mentioned previously in noun phrase extraction, the nested noun phrase '*job listings from other job banks*' is split into two NP candidates, i.e. '*job listings from other job banks*' and '*other job banks*'. But it is unlike to have a coreference relation between them. (b) **Appositive**: if two NPs are in a possible appositive construct, they are most likely to be coreferent. For instance,

E11. *At the national level, America's Career Kit, a suite of Internet application, has been developed by the US Department of Labour*.

In example (E11), '*a suite of Internet application*' is served to define '*America's Career Kit*', and thus they refer to the same entity. Our system determines whether one NP is in appositive to the other NP based on the dependency tree generated by the Stanford Parser[13]. (c) **Syntactic role**: this heuristic favours the NP candidates that have the same subject or object role in the sentence, as in example (E12).

E12. *Pre-Audit inputs revised data into the cost estimate, and stores the data in the system for tracking.*

Here, '*revised data*' (object) is the reference of '*the data*' that is also the object in the sentence.

Semantic Heuristics. This type of heuristics makes use of semantic information (e.g., the structure information among the words). **Semantic class agreement**: If both NPs are associated with the same semantic classes, such as '*Person*', '*Organization*', and '*Location*', it is likely that they are coreferent. The knowledge about the semantic class of a NP is derived based on the hierarchical structure of WordNet[14]. The semantic classes of the NPs, NP_i and NP_j, are in agreement if one is the parent node of the other, or they are in the same node, e.g., '*employee*' and '*worker*'. When one NP has multiple word senses in WordNet, the search will only focus on the first few senses (≤4). Unfortunately, requirements documents often contain acronyms, specific

[11] http://biotext.berkeley.edu/software.html

[12] Proper names are names of persons, places, or certain special things. They are typically capitalized nouns, such as '*London*', '*John Hunter*'

[13] http://nlp.stanford.edu/software/lex-parser.shtml

[14] http://wordnet.princeton.edu/

product names or other domain jargon, so - missing more specific lexical resources - this heuristic cannot be applied extensively.

4.2.2 Building Coreference Classifier

The classifier is built by training a k-nearest neighbour learner with feature sets obtained from the definitions in Table 2. A training instance consists of a pair of NPs which is associated with a class label denoting whether or not the NP pair is coreferential. The k-nearest neighbour learning algorithm is the lazy.IBk algorithm provided by the WEKA package[15]. This was chosen as we found it outperforms other learning algorithms provided by the package. The classifier built on the training dataset has achieved a precision of 82.4% and a recall of 74.2%.

Table 2. Feature set for the coreference classifier. Each instance consists of a pair of NPs, NP_i and NP_j, characterized by 12 features

Feature Type	Feature	Description
String-matching	Full-string matching	Y if both NPs contain the same string after the removal of non-informative words, else N.
	Headword matching	Y if both NPs contain the same headword, else N.
	Modifier matching	Y if both NPs share the same modifier substring, else N.
	Alias name	Y if one NP is the alias name of the other NP, else N.
Grammatical	NP type (NP_i)	Y if NP_i is either definite NP or demonstrative NP, else N.
	NP type (NP_j)	Y if NP_j is either definite NP or demonstrative NP, else N.
	Proper name	Y if both NPs are proper names, else N.
	Number agreement	Y if NP_i and NP_j agree in number, else N.
Syntactic	PP attachment	Y if one NP is the PP attachment of the other NP, else N.
	Appositive	Y if one NP is in appositive to the other NP, else N.
	Syntactic role	Y if both NPs have the same syntactic role in the sentence, else N.
Semantic	Semantic class	Y if NP_i and NP_j agree in semantic class, else N.

Having trained the coreference classifier, it can be applied to instances of anaphoric ambiguity. As described earlier, each instance of an anaphor is associated with a set of candidate antecedents. To identify potential coreference relations among the candidate antecedents, a pairwise comparison of the NPs is carried out using the classifier. In this way, each NP pair can be tested for coreference, and sets of coreferent candidates identified.

Table 3 shows the change of judgment count for an anaphoric ambiguity instance when coreference resolution is considered. The NPs, '*the CCSDS parameters*' and '*these parameters*', are grouped into the same candidate cluster by the trained classifier. Hence, the final judgments of the entity referred by the cluster are, in practice, the combination of the judgments for these two NPs.

[15] http://www.cs.waikato.ac.nz/~ml/index.html

Table 3. The change of judgment count of antecedent candidates with coreference resolution

1. The Raw Data Processing Subsystem first checks the CCSDS parameters for valid values and stores these parameters if **they** are valid.		
Before Coreference Resolution		
	Response Percent	Response Count
(a) the RAW Data Processing Subsystem	0%	0
(b) the CCSDS parameters	50%	7
(c) valid values	0%	0
(d) these parameters	50%	7
After Coreference Resolution		
	Response Percent	Response Count
(a) the RAW Data Processing Subsystem	0%	0
(b) {the CCSDS parameters, these parameters}	100%	14
(c) valid values	0%	0

5. Antecedent Classifier Construction

Nocuous ambiguity occurs when different readers interpret the same text differently. In this section, we attempt to investigate whether heuristics can be developed which model nocuous ambiguity. Our approach has three important elements: collection of human judgements (described previously in Subsection 2.2), developing the heuristics that model those judgments, and a machine learning component to train the heuristics. These elements are represented within the dashed line of Figure 4.

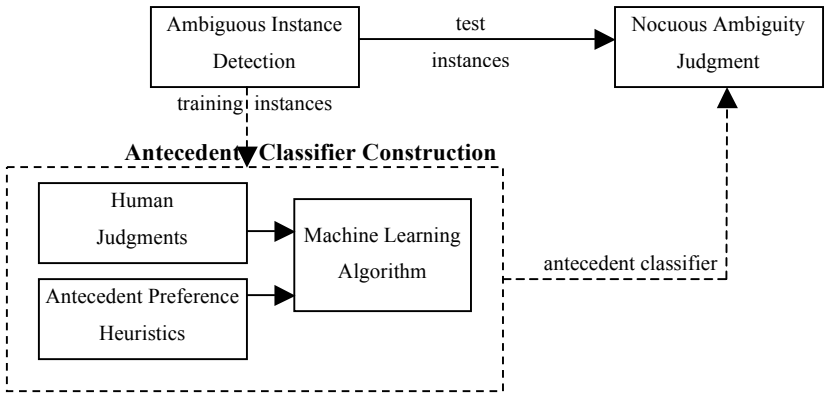

Fig 4. The functional architecture of the antecedent classifier construction module (highlighted by the dashed line)

This section focuses on which heuristics should be used to build an antecedent preference classifier, then section 6 considers how the combined heuristics should be trained. Our anaphora dataset with collected human judgments is used as input to the classifier, with each NP candidate within one anaphora instance being treated as a training instance for learning. The human judgements for each NP are used to determine the preference class label of the NP. A number of antecedent preference heuristics are developed, each of which is regarded as one feature in the feature vector associated with a NP candidate. A set of training instances are fed into a machine learning tool to construct a classifier. Given an anaphor and a set of possible NP antecedents, the

classifier then predicts how strong the preference for each NP is, and from there, whether the ambiguity is nocuous or innocuous.

5.1 Anaphora Preference Heuristics

When applied to anaphora, our model of nocuous ambiguity involves estimating the risk that different stakeholders interpret the anaphor as referring to different antecedents. This is distinct from disambiguation, a process which seeks to determine a unique antecedent, and which hence assumes that there is a single correct interpretation. It is also distinct from identifying when an individual stakeholder experiences a requirement statement as ambiguous: since no stakeholder can be assumed to have access to any other stakeholder's interpretation, he or she cannot judge whether others will interpret the case differently. The purpose of our model is to highlight ambiguity problems that cannot be identified by a single stakeholder alone.

Building on Willis et al. [54], our solution is to estimate the risk of divergent interpretations by applying a collection of heuristic rules, each marking a factor which would favour or disfavour an NP as the antecedent of an anaphor. Heuristics run over the text and their output is recorded in a matrix, which is later used by our machine learning algorithm to classify anaphoric ambiguities as nocuous or innocuous.

Here, we give a brief description of the heuristics we have developed, with some examples. Each heuristic implements a preference factor drawn from the literature on anaphoric ambiguity. The heuristics fall into three types: those related to *linguistic properties* of words and sentence components, those related to *context and discourse* information, and those related to information on how words and sentence components distribute in the language. This distributional information is approximated by *statistical information* drawn from corpora, in this case in English. These heuristics embody a robust, knowledge-rich approach to estimate the preference for candidate antecedents of anaphora. Apart from corpora, we used lexical, syntactic, and semantic knowledge extracted from several linguistic resources including Stanford Parser, WordNet, BNC[16] (British National Corpus), VerbNet[17], and the Sketch Engine[18] [26].

In this paper, we describe 4 new heuristics as well as the original set of 13 heuristics presented in Yang et al. [58]. They are: coordination pattern preference, indicating verb, section heading preference, and domain-specific term preference.

Below we describe our heuristics with some examples. The heuristics generate a collection of features for each candidate NP, summarized in Table 4. These are input to the classifier described in Section 5.2. It is important to understand that all our heuristics mark antecedents with preference factors based on different kinds of knowledge or evidence that favour or disfavour each particular NP for the role of preferred antecedent. None of the heuristics represent obligatory conditions for an antecedent to meet. Rather, we expect the preferred candidate NPs to emerge through the interplay of heuristics which will mutually reinforce or neutralize each other's contribution, as we will see in Section 5.2. This approach has the added advantage that we can

[16] http://www.natcorp.ox.ac.uk/
[17] http://verbs.colorado.edu/~mpalmer/projects/verbnet.html

exercise some control over optimization by adding, or removing, heuristics. Furthermore, the methodology can be extended to cover other ambiguity types by developing heuristics designed to handle them.

5.1.1 Linguistic Preference Heuristics

Our linguistic heuristics record syntactic and semantic clues found useful in identifying antecedents.

H1. Number agreement. In English, anaphors and their antecedents usually must agree in number (i.e. singular or plural). There are exceptions: for instance, certain collective nouns in English (e.g. '*organization*', '*consortium*', '*team*') appear to be singular but can be referred to by a plural anaphor (e.g., '*they'*). Similarly, singular antecedents that have been conjoined with *'and'* or *'or'*, as in (E13), take a plural anaphor:

E13. *IT developer and consultant often ask for an exemplary requirements specification as a starting point in* ***their*** *specific project.*

Finally, plural pronouns are often intended to refer to a singular antecedent to avoid using a gender-specific pronoun. Such instances appear in the requirements documents in our sample and always involve an antecedent that is a single person.

E14. *The user will be able to generate a printout to a local printer from* ***their*** *browser.*

In our system, each noun and pronoun is assigned a number property by the Stanford tagger which is checked by the Number Agreement heuristic. We use a syntactic rule to detect the conjoined NPs and assign the appropriate number property. We handle the exceptions by using the *hypernym* relationship in WordNet, specifically those hypernyms which point to collective nouns referring to groups such as '*staff*' and '*division*', or to descriptions of individuals like '*employee*', or '*user*'. Moreover, we make use of a list of expressions that modify plural nouns (e.g., '*a lot of*' and '*a few*', '*many*') in assigning the number property.

H2. Definiteness. A definite noun phrase is more likely to be an antecedent of an anaphor than an indefinite NP [35]. The Definiteness heuristic tags an NP as definite if its head noun is modified by a definite article (e.g., '*the system*'), a demonstrative (e.g., '*this function*'), or a possessive pronoun (e.g., '*its attribute*').

***H3. 'Non-prepositional' noun phrases*.** Brennan et al. [7] showed that a priority system operates between syntactic roles in determining antecedents. In English, the priority order, implemented by our heuristic, is '*subject*, *direct object*, *indirect object'*. In English, subjects and direct objects are not introduced by prepositions (such as *in, to, up*, etc.) and our heuristic favours non-prepositional NPs as antecedents. For instance:

E15. *Constraints are conditions about the data that must always be true.* ***They*** *are the integrity rules that protect the data in the eventual database.*

---

[18] http://sketchengine.co.uk/

Here, the NPs '*constraints*' and '*conditions*' are more likely antecedent candidates because '*the data'* is part of the preposition phrase '*about the data*'.

H4. Syntactic constraint. Empirical evidence suggests that the syntactic roles of antecedent and anaphor often correspond [11]. This heuristic gives preference to candidates that have an identical subject or object role as the anaphor. For example:

E16. *The VCDUs are annotated to reflect the data quality, and any change in the VCID.* ***They*** *are also annotated to mark the end of the contact period.*

Here '*They*' (the subject) refers to '*The VCDUs*' which is also the subject of its sentence.

H5. Syntactic parallelism. Repeating patterns may offer clues about preferred antecedent candidates. For instance, in (E17), both '*This CONOPS*' and '*it*' are the subjects in their own sentence, and are followed by the verb '*describes*'. This heuristic favours *'This CONOPS'* as an antecedent.

E17. *This CONOPS [describes] the mission of the LMI system.* ***It*** *also [describes] the functions and characteristics of the system.*

H6. Coordination pattern. This heuristic marks a preference for candidates that are in the same location as the pronoun in a coordination pattern. Application is restricted to the patterns, '[*Subj.*] V_1 *NP conj* V_2 *it/them*', where conj = {*and*, *or*, *before*, *after*}. For instance:

E18. *This subsystem is responsible for [receiving] raw wideband data and [placing]* ***it*** *in a datastore for RDPS processing.*

In (E18), the verbs, '*receiving*' and '*placing*', are coordinated by the conjunction '*and*', and '*raw wideband data*' and '*it*' are separately the corresponding objects of the coordinated verbs. Therefore, the heuristic prefers the NP, '*raw wideband data*' as the antecedent for the anaphor, '*it*'.

H7. 'Non-associated' noun phrases. This heuristic disfavours NPs occurring immediately with the anaphor but in a different syntactic role. For example:

E19. *The technical interfaces to the EHR system must be documented in such a way that the customers can understand* ***them*** *and use* ***them*** *for integration.*

In (E19), the NP '*the customers*' is unlikely to be the antecedent because it stands in a subject relation to the same verb '*understand*' of which '*them*' is the object.

H8. Indicating verbs. Empirical evidence suggests that some verbs, such as '*represent*', '*identify*', '*discuss*', '*describe*', etc., are particularly good indicators of the antecedent. The first NP following them is marked as the preferred antecedent, as shown in (E20). We use a verb list based on Mitkov [35].

E20. *The LPS operational scenarios [represent] sequences of activities performed by operations personnel as* ***they*** *relate to the LPS software.*

H9. Semantic constraint. This heuristic marks a preference for candidate antecedents that respect semantic constraints. For instance, the verb *'to live'* normally requires an animate agent. In example (E21) our heuristic prefers *'many individuals'* over *'resources and services'*.

E21. *Many individuals have difficulty accessing resources and services because* ***they*** *[live] in a geographically remote area.*

Semantic properties and constraints, such as animacy, are collected from WordNet and VerbNet. In the case of E21, 'they' is an anaphor and is a subject that is followed by the verb 'live'. 'live' (whose subject is *they*, the anaphor) has a thematic role as an *animated* verb according to VerbNet. WordNet also gives the NP candidate 'individual' as belonging to the parent node, 'people', which WordNet identifies as an animated noun. Because WordNet and VerbNet agree on the animacy property, the noun satisfies the requirement of an animated verb.

***H10. Semantic parallelism*.** This heuristic records a preference for candidate antecedents sharing a semantic role with the anaphor (e.g., agent) as below:

E22. *RDCS [captures] a raw data byte stream after* ***it*** *[receives] the start capture directive from the MACS.*

This heuristic plays a particularly useful role since requirements documents contain many domain-specific terms (e.g., '*RDCS*') that are not present in WordNet and for which semantic properties cannot be retrieved. Verbs, on the other hand, are far less likely be highly domain specific, and in example (E22), both '*captures*' and '*receives*' are tagged as requiring agents in VerbNet and the heuristic allows us to use this constraint to record a relationship between '*RDCS*' and '*it*'.

H11. Domain-specific term. NPs that represent domain specific terms are more likely to be the antecedent than non-domain specific NPs. For this heuristic, we collect domain-specific term lists from two types of sources: (a) term dictionaries such as controlled vocabulary list, glossaries, and index lists, when contained in the requirements document; (b) Acronym or abbreviation list (see the discussion of 'String-matching Heuristics' in Section 4.2.1).

5.1.2 Context/Discourse Preference

The overall performance of antecedent determination can be improved when incorporating features that capture context/discourse information [7, 35]. These are distinct from the syntactic and semantic clues discussed so far, and concern what is best described as the structure of the *information flow* in a document. We have developed four heuristics.

H12. Centering (discourse focus). This heuristic attempts to capture the most salient entities in a connected chunk of document (e.g., a paragraph or section/subsection) by marking a preference for the most frequently occurring candidate antecedents. We observed that about 87% (10,037 out of 11,552) of paragraphs in the dataset contain fewer than 3 sentences, which implies that the occurrence frequency of a NP is unlikely to be high.

We compared the occurrence information of potential antecedents in both the paragraph in which they appeared, and the number of times they appeared in the whole document. We found that a reliable predictor for frequently occurring candidates was that the candidate noun would occur at least twice in the same paragraph as the anaphor. Therefore, this heuristic gives a preference for antecedent candidates that occurred twice or more in the paragraph.

H13. Section heading. If a NP candidate also appears in the heading of the section in which the ambiguous instance is located, then we mark it as a preferred candidate. The section heading

information is extracted from the '*Table of Content*' or '*Content*' page at the beginning of the document, or using a simple string matching pattern, i.e., section headings usually start with a number and a dot followed further numbers/dots (e.g., '*1.1*', '*2.1.2*'). For example, the section title for the paragraph where Example (E22) appears is '*4.4.1 Functional Requirements allocated to the RDCS*', which contains the NP candidate '*RDCS*' of interest.

H14. Sentence recency. Antecedents and anaphora may occur in different sentences. This heuristic gives preference to candidate antecedents that occur in the same sentence as the anaphor. In Example (23), the NP candidate, '*each relationship*', is more likely to refer to the pronoun '*it*' than other NP candidates that appear in the previous sentence.

E23. *The relationships represent the association between the instances of one or more entities that are of interest to the LPS. For each relationship, there is a cardinality associated with* ***it****.*

H15. Proximity. The distance between antecedent and anaphor is significant and this heuristic allocates a rank to candidate antecedents reflecting their relative positions in the left context of the anaphor. Candidates closer to the anaphor are ranked higher (see Table 4). The distance of a NP candidate to an anaphor is the number of words between the right end of the NP and the anaphor itself. For example, in E18, the distances of the NPs, '*This subsystem*' and '*raw wideband data*' from the pronoun '*it*' are 10 and 2 respectively. The heuristic prefer '*raw wideband data*' is an antecedent.

5.1.3 Statistics/Corpus Heuristics

Although requirements documents tend to be highly specialized and contain terms that are not part of the general English vocabulary, comparison of textual features with a balanced sample of standard text, for instance drawn from a general corpus, can provide useful insights. For instance, example (E21) has two candidate antecedents (*'many individuals'* and *'resources and services'*) for the pronoun *'they'* which is subject to the verb *'live'*. Evidence that one of the two candidates co-occurs frequently with the verb not just in the document itself, but also in other contexts increases the possibility that readers will assume it is the antecedent.

H16. Local-based collocation frequency. This heuristic marks a preference for candidate antecedents which appear in co-occurrence patterns in the requirements document itself.

E24. *This subsystem provides the functionality to synchronize the major frames, extract major frame times, deinterleave band data, reverse band data if necessary, and align band data.* ***It*** *is also responsible for generating the Calibration and Mirror Scan Correction files.*

In Example (E24), unlike other NP candidates, the word '*subsystem*' frequently co-occurs with the adjective '*responsible*' in the document, and thus '*this subsystem*' is more likely to be considered as the potential antecedent.

H17. BNC-based collocation frequency. This heuristic marks a preference for candidate antecedents which appear in co-occurence patterns in a large corpus: in this case, the British National Corpus (BNC). The BNC is a modern corpus of written text containing over 100 million words of English, collated from a variety of sources, including some from the same domains as the

documents in our corpus. We use the Sketch Engine to collect information about the behavior of words and phrases, in the form of collocation frequencies and patterns, from the BNC. For instance, in Example (19), both headwords '*interface*' and '*system*' have a collocation score of 2.77 and 7.23 with the verb '*use*' in the '*object-of*' relation of the BNC, whereas '*way*' and '*customer*' has no such collocation relation with '*use*'. Therefore, the heuristic prefers the NPs '*The technical interfaces*' and '*the EHR system*' to the NPs '*a way*' and '*the customers*'.

Table 4. Features for the antecedent classifier. Each instance represents a single NP candidate, NP_j, characterized by 17 features.

Feature Type	Feature	Description
Linguistics	Number agreement	Y if NP_j agree in number; N_P if NP_j does not agree in number but it has a person property; N if NP_j doesn't agree in number; UNKNOWN if the number information cannot be determined.
	Definiteness	Y if NP_j is a definite *NP*; else N.
	Non-prepositional NP	Y if NP_j is a non-prepositional NP; else N.
	Syntactic constraint	Y if NP_j satisfies syntactic constraint; else N.
	Syntactic parallelism	Y if NP_j satisfies syntactic parallelism; else N.
	Coordination pattern	Y if NP_j satisfies coordination pattern; else N.
	Non-associated NP	Y if NP_j is a non-associated NP; else N.
	Indicating verb	Y if NP_j follows one of the indicating verbs; else N.
	Semantic constraint	Y if NP_j satisfies semantic constraint; else N.
	Semantic parallelism	Y if NP_j satisfies semantic parallelism; else N.
	Domain-specific term	Y if NP_j is contained in the domain-specific term list; else N.
Context	Centering	Y if NP_j occurs in the paragraph more than twice; else N.
	Section heading	Y if NP_j occurs in the heading of the section; else N.
	Sentence recency	INTRA_S if NP_j occurs in the same sentence as the anaphor; else INTER_S.
	Proximal	integral value *n*, where *n* means that NP_j is the *n*th *NP* to the anaphor in the right-to-left order
Statistics	Local-based collocation frequency	integral value *n*, where *n* refers to the occurrence number of the matched co-occurrence pattern containing NP_j in local requirements document
	BNC-based collocation frequency	Y if the matched co-occurrence pattern containing NP_j appears in the word list returned by the sketch engine; else N.

5.2 Building a Machine-Learning Based Antecedent Classifier

The heuristics described in Section 5.1 are antecedent *preferences* and not obligatory conditions. There might be cases where one or more of the antecedent preferences do not 'point' to the correct antecedent. When all preferences are taken into account, however, the preferred NP candidate is still very likely to be traced down. Moreover, we can easily add more new heuristics or delete some unimportant heuristics if it is needed in order to optimize the system performance.

Indeed, individually, the heuristics have limited predictive power: their effectiveness lies in their ability to operate in concert. We try to harness this by using machine learning (ML) techniques to combine the outputs of individual heuristics. ML is an area of computer science

concerned with attempting to recognize complex patterns automatically and make intelligent decisions based on empirical data. Many ML algorithms allow learning of complex and nonlinear relations between heuristics which display complex interdependencies, such as those described in section 5.1.

The ML techniques used in our system derive from supervised learning, whereby a function is inferred from a set of annotated training data, to classify instances of ambiguity into nocuous or innocuous instances. The training data for the antecedent classifier consists of a set of NP candidate instances, each of which contains a feature vector that is made up of heuristics scores (described in Table 4) and a class label that is determined by the distribution of human judgments as captured by thresholds, explained below.

Classification is performed on a discrete-valued feature space. In other words, each training/test instance (i.e. an NP candidate) is represented as an attribute-value vector, where attributes describe properties of the antecedent preferences mentioned earlier (see Table 4) and values the outcomes associated with the corresponding heuristics.

The class label associated with each NP training instance is marked as one of the three antecedent categories, *positive (Y)*, *questionable (Q)*, or *negative (N)*. These are obtained from the distribution of judgment responses collected from the anaphora surveys. Class labels in the training set are associated with a particular ambiguity threshold τ. Specifically, in an innocuous ambiguity case, only one NP candidate can be marked as *positive* (Y) when the percentage of the judges selecting it as the correct reference is above the ambiguity threshold τ. The remaining NPs are tagged as *negative* (N). On the other hand, in a nocuous ambiguity case, an NP is labeled as *questionable* (Q) only if the percentage selecting it is below τ but greater than 0. Otherwise, it is tagged as *negative* (N). So if a particular anaphor had, for example, three potential antecedents, then that would give three training cases.

Tables 5 and 6 illustrate how class labels for antecedent candidates in a nocuous and an innocuous ambiguity case are calculated. Table 5 illustrates a case of *nocuous* ambiguity at threshold τ=0.8. Antecedent candidates (a) and (b) are labeled as '*Q*' because both attracted some preference judgments but no candidate score exceeded the threshold. Table 6 shows a case of *innocuous* ambiguity at threshold τ=0.8. Candidate (c) is tagged with '*Y*' because its response score exceeds the threshold. The others are tagged with '*N*'.

Table 5. Determining the class label for antecedent candidates in a *nocuous* ambiguity case at threshold τ=0.8

1. The LPS operational scenarios represent sequences of activities performed by operations personnel as **they** relate to the LPS software.		
	Response Percent	Class Label
(a) the LPS operational scenarios	33.3%	Q
(b) sequences of activities	66.7%	Q
(c) activities	0%	N
(d) operations personnel	0%	N

Table 6. The determination of antecedent label for the NP candidates in an *innocuous* ambiguity case at threshold τ=0.8

2. Testing performed to demonstrate to the acquirer that a CSCI system meets **its** specified requirements.		
	Response Percent	Class Label
(a) Testing	0%	N
(b) the acquirer	16.7%	N
(c) a CSCI system	83.3%	Y

As mentioned before, we use a coreference resolution engine to link possible coreferring NPs within a list of antecedent candidates for a given anaphor. Table 7 shows the impact of coreference resolution on the class labels associated with the list of candidate antecedents. The effect of the process is to conflate the judgments associated with coreferring NPs, which in turn affects the distribution of judgments, and their relation to the threshold. In this case, the NPs '*the CCSDS parameters*' and '*these parameters*' were identified as coreferring and the sum of the judgment proportions associated with both exceeds the threshold – in this case τ=0.8. Consequently, the ambiguity instance in Example (3) is judged as *innocuous*, and the classes label *'Y'* is assigned. Note that without coreference resolution, the instance would have been considered nocuous.

Table 7. The determination of antecedent label for the NP candidates in an ambiguity case with *coreferring* NPs at threshold τ=0.8

3. The Raw Data Processing Subsystem first checks the CCSDS parameters for valid values and stores these parameters if **they** are valid.		
Before Coreference Resolution (Nocuous)		
	Response Percent	Class Label
(a) the RAW Data Processing Subsystem	0%	N
(b) the CCSDS parameters	50%	Q
(c) valid values	0%	N
(d) these parameters	50%	Q
After Coreference Resolution (Innocuous)		
	Respond Percent	Class Label
(a) the RAW Data Processing Subsystem	0%	N
(b) {the CCSDS parameters, these parameters}	100%	Y
(c) valid values	0%	N

The antecedent classifier is built using the Naive Bayes algorithm within the WEKA machine learning algorithm package. It is designed to maximize the accuracy of prediction for antecedent class labels. The training data is obtained from the dataset of anaphoric ambiguity instances and the judgments associated with each possible antecedent candidate. Table 8 shows the distribution of the three antecedent class labels for the collection, at different thresholds and after coreference resolution. As expected, as the threshold τ increases, the number of *Questionable* class labels increases, and the number for both *Positive* and *Negative* instances decreases accordingly.

Table 8. Distribution of three types of antecedent instances at different ambiguity thresholds

	Antecedent Class Label (%)		
	Y	**Q**	**N**
τ = 0.50	21.20	6.24	72.6
τ = 0.55	19.67	9.89	70.4
τ = 0.6	19.19	10.83	69.96
τ = 0.65	17.20	15.19	67.61
τ = 0.70	16.49	16.73	66.78
τ = 0.75	15.07	19.78	65.15
τ = 0.80	12.95	23.80	63.25
τ = 0.85	9.66	29.45	60.89
τ = 0.90	9.31	30.03	60.6
τ = 1.00	4.94	36.40	58.66

6. Nocuous Ambiguity Judgment

As described previously, we used a machine learning algorithm to construct an automated tool - an *antecedent classifier* – that, given a pronoun and a candidate NP antecedents, assigns a weighted antecedent tag to the NP candidate. The antecedent tag information in turn is used by the tool to predict whether the anaphora instance displays nocuous ambiguity. We compare automatic judgments returned by the system with the collected human judgments to evaluate system performance. Figure 5 shows the functional architecture of the Nocuous Ambiguity Judgment module, which is highlighted by the dashed line.

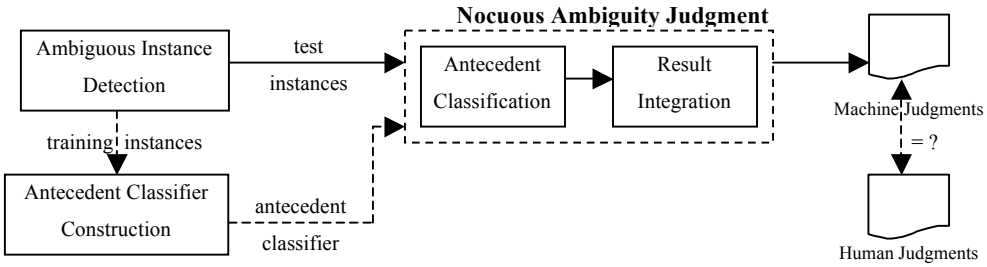

Fig. 5 The functional architecture of the Nocuous Ambiguity Judgment module (highlighted by the dashed line)

This module would find a natural home as a component of a requirements authoring environment[19], where it would serve (in a role similar to that of a spell-checker in a word processor) to identify and highlight cases of nocuous ambiguity as soon as the requirement is written. We believe that rather than attempt to rewrite the text into a form that is less likely to be misunderstood, the author of the requirement, at the very time when the requirement is committed to written form, is best placed to rewrite it, based on knowledge of the intended meaning. As no one author is able to assess whether a particular ambiguous instance is likely to be misunderstood by other readers, our vision is for a tool which *notifies* authors that their text may lead to

[19] In fact, as part of our future work we intend to integrate the technology in the popular DOORS requirements management tool.

misunderstandings. This allows the author to retain control of the requirements writing process, and to deal with potentially problematic cases in a manner which he or she considers most appropriate,,based on knowledge of who the other stakeholders are, and knowledge of his or her own goals.

This scenario, however, is not the only one possible. In fact, the same technique can be used post-hoc on a whole requirements document, as part of a validation or quality assessment effort, in a similar spirit to Gnesi et al [18] (but with fewer false positives and thus increased precision), or in a stand-alone tool, as part of an acceptance test in a scenario in which requirements writing has been outsourced to a consulting company. In this latter role, our Nocuous Ambiguity identification technique could serve as a way to enforce parts of a given style guide on language used to document requirements.

6.1 Antecedent Classification

To determine the antecedent preference of an NP candidate in a test ambiguity case, we create a test instance in form of a feature vector for the NP, and present the feature vector to the generated antecedent classifier, which returns a class label of *Positive* (*Y*), *Questionable* (*Q*), or *Negative* (*N*).

6.2 Result Integration for Nocuous Ambiguity Identification

As previously discussed, our work emphasizes the discovery of potential nocuous ambiguity in order to notify a user rather than attempting to disambiguate. The user retains control over the final determination based on some contextual analysis. Berry et al. [3] suggest that, in order to improve the user's trust in an ambiguity tool, it is desirable that the tool should find every instance of a particular type of ambiguities (i.e. 100% recall) even at the expense of some imprecision. In line with this high recall principle, the resultant algorithm is designed to maximise recognition of nocuous ambiguity, even at the expense of returning more innocuous cases.

In order to obtain more potentially questionable instances of antecedent prediction, we relax the constraints and introduce two concepts, the weak positive threshold W_Y and the weak negative threshold W_N. The rationale for using weak thresholds is that antecedent preference reflects a spectrum with *Y* (high), *Q* (medium), and *N* (low). Weak positive and negative thresholds act as buffers to the *Q* area. Antecedent NPs that fall in the W_Y or W_N buffer area are treated as possible false negative (FN) for the classification of the label *Q*. The antecedent tags *Y* and *N* are labeled as weak positive or weak negative depending on these thresholds. Figure 6 shows how an ambiguity is classed as nocuous or innocuous when the weak positive and weak negative labels are used. We treat as innocuous those cases where either:

1. the antecedent label list either contains one clear *Y* candidate, whose certainty exceeds all others (Q or N) by a margin, or
2. contains no *Y* candidate, one *Q* candidate, and one or more *N* candidates where the *N* candidates are not *weak negative*.

Determining whether ambiguity is nocuous or innocuous

Given an anaphoric ambiguity instance with multiple potential NP candidates, the antecedent classifier returns an antecedent label set, $C \in \{Y, Q, N\}$, with the corresponding prediction scores, $R = \{r_1, r_2, \ldots, r_n\}$, for individual NP candidates.

Parameters:

1) W_Y - the threshold for the *weak positive* label. The label *Y* is viewed as *weak positive* when the positive prediction score $r_i < W_Y$

2) W_N - the threshold for the *weak negative* label. The label *N* is viewed as *weak negative* when the negative prediction score $r_i < W_N$

Procedure:

if the label list *R* contains

(one *Y*, no *Q, one or more N*)

or

(no *Y*, one *Q*, one or more *N* but not *weak negative*)

or

(one *Y* but not *weak positive*, any number of *Q or N*)

then

the ambiguity is INNOCUOUS

else

the ambiguity is NOCUOUS

Fig. 6 The procedure for determining whether anaphoric ambiguity is nocuous or innocuous

7. Experiments and Results

For the purpose of evaluation, we used a standard 5-fold cross validation technique in which, in each iteration, we trained on 80% and tested on 20% of the remaining data. The performance of the task was measured in terms of precision (P), recall (R), F-measure (F), and Accuracy:

$$R = \frac{TP}{TP + FN} \qquad P = \frac{TP}{TP + FP} \qquad F_\beta = \frac{(1 + \beta^2)PR}{\beta^2 P + R} \qquad Accuracy = \frac{TP + TN}{total}$$

where *TP* (true positives) is the number of correctly identified *nocuous* ambiguities, FN (false negatives) is the number of nocuous ambiguity not identified by the system, and FP (false positives) is the number of nocuous ambiguities y that are incorrectly identified. Here we use the F_2 measure (β=2), which weights recall twice as much as precision in order to emphasize the impact of the recall on performance. All results were averaged across five iterations.

As described earlier, the anaphoric ambiguity model introduced in this paper is an improvement over our original model in Yang et al. [58]. In this paper, we consider the additional effects of using NP coreference information resolution engine and four more antecedent preference heuristics, i.e. coordination pattern preference, indicating verb, section heading preference and domain-specific term preference. We carry out a new set of experiments, and compare *the extended ambiguity model* (**AM_Ext**) with *the original model* (**AM_Ori**) for the purpose of the evaluation of system performance.

We present our experiments and results as follows: First we conducted a set of experiments to compare the performance of the antecedent classifier at different ambiguity thresholds. These are discussed in Section 7.1. Then in Section 7.2, we discuss the impact of various feature types on

performance of the antecedent classifier. The tool is then compared to a baseline measurement, and in Section 7.3, we report on the performance at different ambiguity thresholds.

7.1 Performance of the Antecedent Classifier

The performance of our antecedent classifier trained on the anaphoric instances is illustrated in Figure 7. Since all of the three antecedent categories, *Y*, *Q*, and *N*, are important in the later step of result integration, here we report the accuracy, which we think can better reflect the overall system performance. The best performance of the extended classifier, AM_Ext, was achieved at the threshold of 0.5 with an accuracy of 81.42%, whereas the worst performance was obtained at the threshold of 1.0 with a drop of 7.1% in accuracy (to 74.3%). Observing that at a threshold of 1.0, 80% of ambiguities are deemed nocuous based on human judgment (see Figure 1), the classifier behaviour's is worse than that of the trivial classifier that always answers 'nocuous' for this case. A likely reason for the drop in accuracy at high thresholds is the problem of imbalanced distribution among different types of training instances for high thresholds (e.g., $\tau \geq 0.9$), especially for positive instances which are significantly decreased with the increase of the threshold. For example, there are only 42 positive instances compared with the total of 585 instances at the threshold of 1.0 (see Figure 1).

The AM_Ext classifier performs generally better than the AM_Ori one with a slight average increase of 2.43% in accuracy. This means that the introduction of the NP coreference resolution engine and four new-added heuristics does capture some characteristics of anaphoric ambiguity, and thus improves the performance of the antecedent classifier. However, the effect is not evident. One possible explanation is: (a) The anaphora instances in which two NPs are co-referred occur infrequently in the dataset, and thus the effect of noun-phrase resolution is limit in system performance. (b) Compared with the previous 13 heuristics, the four newly-added heuristics do not capture some important characteristics in anaphora ambiguity.

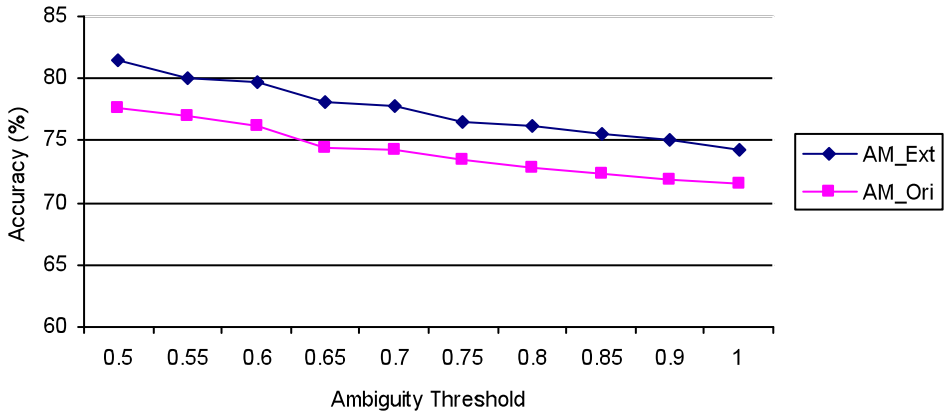

Fig. 7. The performance of antecedent classifiers at different thresholds

7.2 Impact of Feature Types

We also evaluated the impact of individual feature types on system performance compared with the full-fledged antecedent classifier with all features. We treat linguistic features as the basic feature type because of the important role that linguistic information plays in anaphora analysis. Table 9 shows that the context features or statistical features generally improve the performance of the antecedent classifier when they are incorporated with the linguistic features. Nevertheless, it seems that both types of features appear to have no apparent positive influence on antecedent prediction. It suggests that the syntactic and semantic characteristics inherent in the sentences provide the most important information to capture the associations between the antecedent candidates and the anaphor. The contextual and corpus-based statistical information is helpful to some extent, but not fundamental in the analysis of anaphoric ambiguity.

We hypothesized that the best accuracy on all features would be achieved if each type of features could capture disparate characteristics of anaphora to some extent. The results in Table 9 support this hypothesis: the system performs best when all features are used together which results in an increase of 2.1% in accuracy.

Table 9 The impact of feature types on classifier performance (τ=0.8)

Feature Type	# Correct Instance	# Incorrect Instance	Accuracy (%)
Linguistics	630	219	74.18
Linguistics + Context	641	208	75.56
Linguistics + Statistics	635	214	74.84
All Features	647	202	76.25

Table 10. The impact of individual heuristic feature on classifier performance (τ=0.8)

Feature Type	**Heuristic Feature (Excluded)**	**Accuracy (%)**
Linguistics	Number agreement	75.64
	Definiteness	75.48
	Non-prepositional NP	75.72
	Syntactic constraint	**72.28**
	Syntactic parallelism	75.86
	Coordination pattern	76.05
	Non-associated NP	75.62
	Indicating verb	75.83
	Semantic constraint	**74.57**
	Semantic parallelism	76.02
	Domain-specific term	76.02
Context	Centering	75.86
	Section heading	76.02
	Sentence recency	75.14
	Proximal	75.86
Statistics	Local-based collocation frequency	75.74
	BNC-based collocation frequency	75.80
	All features	**76.25**

Moreover, we conducted a set of experiments to evaluate the impact on system performance with respect to individual heuristics. In each run, one of the heuristics was *excluded* from the

feature set. The comparison of classifier performances for different heuristics (shown in Table 10) indicates that the heuristics, syntactic and semantic constraint, are more important features in the prediction of preferred antecedent, which contribute 3.7% and 1.68%, respectively, on accuracy improvement. However, some heuristics, such as coordination patterns, semantic parallelism, domain-specific terms, and section heading, display trivial influence on accuracy performance. Another interesting observation is that there is no one heuristic strong enough to make a decisive prediction. This evidence corroborates our findings in previous work [9], that the classifier performs well only when multiple heuristics are taken into consideration, even if the heuristics appear to be poor predictors individually.

7.3 System Performance Comparison

7.3.1 Comparing with the Baseline Model

We use a baseline model to compare the performance of the proposed ML-based model for nocuous ambiguity identification. In the baseline model, we assume that each recognized ambiguity instance has the potential to be a *nocuous* ambiguity, and is counted as a positive match for the baseline model. Therefore, the baseline model achieves an 'ideal' recall R_{BL} of 100%, and the precision and F-measure are calculated as:

$$P_{BL} = \frac{\#\,Nocuous\ identif\ edby\ Judgments}{total\ \#\ of\ Ambiguities}$$

$$F_{BL} = \frac{(1+\beta^2)P_{BL}R_{BL}}{\beta^2 P_{BL} + R_{BL}} (\beta = 2)$$

The performance comparison among the ML-based model with weak thresholds, the ML-based model without weak thresholds, and the baseline model at different ambiguity thresholds are given in Table 11. As expected, compared with the baseline model, the ML-based model with weak thresholds performs better with an average increase of 7.79 percentage point on precision and 6.93 percentage points on F2 measure, although the recall is slightly lower compared with the ideal of 100%. However, for the ML-based model without the weak thresholds, we see notable improvement in precision, but much larger decreases in recall. Overall, the results suggest that the ML-based model benefits from using antecedent preferences information represented in the antecedent classifier.

Comparing the ML-based model without weak thresholds, we see notable gains in recall, and smaller drops in precision for the ML-based model with weak thresholds. It reveals that the introduction of the weak positive and negative thresholds help catch more instances that fall into the grey areas between positive and questionable instances, or between questionable and negative instances. Those instances are very sensitive to the ambiguity threshold τ.

Furthermore, we also find that the overall performance of the system begins to deteriorate as the threshold τ decreases, especially for precision and F-measure. As mentioned before, our technique is optimized to prefer high recall, at the expense of lower precision. In order to find more nocuous ambiguities, the determination of innocuous ambiguities is quite strict (as shown in procedure in Figure 6). However, at low thresholds, more ambiguities are considered innocuous,

because fewer readers are required to agree on a common interpretation. But the system requirement of high recall results in more false positives (innocuous cases being classed as nocuous) at low thresholds. This results in lower precision and F-measure at these thresholds.

Table 11. The performance comparison between two ML-based models and the baseline model

	ML-based Model (AM_Ext_1) (with weak thresholds)			ML-based Model (AM_Ext_2) (without weak thresholds)			Baseline Model		
	P	R	F2	P	R	F2	P	R	F2
$\tau = 0.5$	0.2405	0.9500	0.5974	0.2724	0.5500	0.4568	0.1000	1.0	0.3571
$\tau = 0.55$	0.2989	0.9722	0.6702	0.3182	0.6745	0.5510	0.1950	1.0	0.5477
$\tau = 0.6$	0.3091	0.9750	0.6814	0.3385	0.6750	0.56305	0.2150	1.0	0.5779
$\tau = 0.65$	0.3760	0.9825	0.7428	0.3871	0.7368	0.62404	0.3000	1.0	0.6818
$\tau = 0.7$	0.4388	0.9841	0.7881	0.4506	0.7619	0.6694	0.3300	1.0	0.7112
$\tau = 0.75$	0.4632	0.9867	0.8047	0.4667	0.8400	0.7241	0.3900	1.0	0.7617
$\tau = 0.8$	0.5614	0.9785	0.8548	0.5802	0.8495	0.7773	0.4800	1.0	0.8219
$\tau = 0.85$	0.6485	0.9833	0.8912	0.6524	0.8717	0.8167	0.6100	1.0	0.8866
$\tau = 0.9$	0.6737	0.9837	0.9008	0.6906	0.8862	0.8386	0.6250	1.0	0.8928
$\tau = 1.0$	0.8201	0.9937	0.9533	0.8216	0.9497	0.9209	0.8000	1.0	0.9523

7.3.2 Comparing with the Original Ambiguity Model

Figure 8 shows the system comparison of the extended ambiguity model, AM_Ext, with the original one, AM_Ori, at different ambiguity thresholds. Although the AM_Ext model merely outperforms the AM_Ori model on F-measure with a small average increase of about 1 percentage, it is interesting to note that the apparent improvement takes place at the threshold range between 0.55 and 0.75. One possible explanation is due to the class skewness problem, that is, the lack of the *Questionable* instances at the lower thresholds (τ <0.55) and the lack of the *Positive* instances at the higher thresholds (τ>0.75) (See Figure 1). The ML-based classifier could not perform well because of the unbalanced distribution of instances types at the lower or higher thresholds.

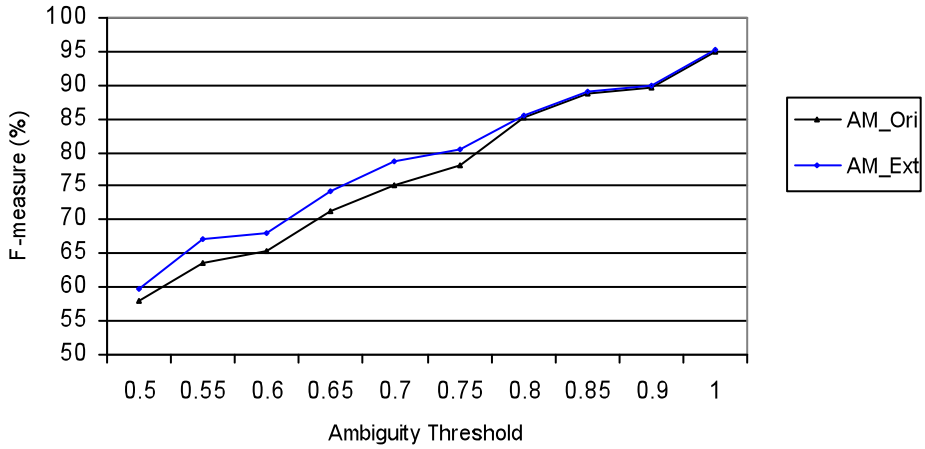

Fig. 8. The performance comparison between the extended model and the original model

8. Threats to Validity

This section describes the potential threats that might affect the validity of our work and the nocuous ambiguities identified by our tool, and discuss how they are mitigated or accommodated.

Context Information. As mentioned earlier, anaphoric ambiguity is a somewhat context-dependent ambiguity in which contextual information (e.g., discourse focus, the header of the text) provides some useful clues for interpreting the sentence. Structured documents (e.g., HTML and XML files) denote structural semantics for text such as title, headings, paragraphs, and other items. In fact, while many requirements documents are managed as structured databases (e.g., in tools such as DOORS), many others are distributed in PDF or word-processing format, especially when crossing inter-organizational borders. Recovering useful structure information for analysis from these formats can be a substantial challenge. Moreover, the relationships between requirements (e.g., prerequisite, supports, extends, conflicts), when available, could also be utilized when making judgments.

In our surveys, judges were presented only with individual anaphora instances without the surrounding context. This can lead to additional difficulties for the judges in deciding on the correct interpretation, and is thus a threat to how generalizable to RE practice our results are. For example:

E25. *LVL1 trigger accept Identifier, A 24 bit L1ID is provided from the TTCrx with each LVL1A signal. In conjunction to the BCID, **it** uniquely defines an event.*

In this example, more than a half of the judges (7 out of 13 judges) could not decide which NP candidate is most likely to be the antecedent of the anaphora with respect to the five NP candidates that appear in the preceding text. Therefore, for most thresholds, this instance exhibits nocuous ambiguity. However, the header information of this paragraph was '*L1ID*'. Had the judges been provided with this, it is possible that this would have affected their choice of antecedent.

This is particularly relevant for requirements documents. In fact, not only are requirements documents often highly structured, with numerous levels of headings and sub-headings which contribute to scope and qualify the actual text very accurately, but the potential readers of the document are generally familiar with the application domain, and can draw on very specific knowledge in order to select the most plausible interpretation.

Nevertheless, the lack of contextual information means that, in our experiment, certain instances were judged as nocuous whereas, if presented in-context and to informed judges, they would have turned out to be innocuous. In essence, we risk having underestimated the number of false positives - which, as already discussed, is considered in the literature a minor problem.

Domain-specific Corpus. Corpus-based statistics information is one of the main resources used by our heuristics. The corpus we used in our studies was the BNC corpus, which is a large *generic* text corpus. However, using a domain-specific corpus might improve the heuristics' performance, especially for domain-specific ambiguities. If, as expected, training the classifier on a domain-specific corpus (which, however, would have to be different for each project or product family) yields better results, the effect of this threat is that we have again underestimated the

performance of our approach. Another consideration is also in order: in requirements engineering, it is often the case that different stakeholders have different areas and degrees of domain expertise. For example, an analyst might not have the specific knowledge possessed by a domain expert, or the latter might not share the specialized language of the developers. Hence, training a classifier on a highly-specialized domain corpus could be detrimental, in that ambiguities that could be dangerous for some of the stakeholders, would be deemed innocuous on the ground that other stakeholders (who have highly-specialized domain knowledge) would not consider them ambiguous. We therefore believe that using a generic corpus is appropriate in requirements engineering.

Machine Learning (ML) Approaches. To select an appropriate machine learning algorithm to build our antecedent classifier, we tested a number of ML algorithms available in the WEKA package on our dataset. The reason we selected the Naive Bayes algorithm in our tests was because that algorithm generally performed better than other candidates at various levels of ambiguity thresholds. Naive Bayes achieved an average accuracy of 73.6% compared with Decision tree (70.39%), J48 (71.4%), LogitBoost (72.09%), and SVM (70.16%). However, during training, we found that the performance of a variety of ML algorithms on our dataset was relatively stable. Most of the ML algorithms achieved similar performances, with accuracy ranging between 70% and 74%. We conclude then that the particular choice of a ML algorithm would not substantially change our results.

Still, it would be interesting to investigate whether the performance of our classifier can be improved by using other techniques such as Neural Networks or Genetic algorithms.

Threshold settings in requirements engineering. The objective of our research is to develop techniques to identify the sentences containing nocuous ambiguities that carry high risk of misunderstanding. What is “high risk” is determined in our technique by a threshold, and having set the wrong threshold in our experiments is a threat to the technique's applicability in other situations. When our technique is applied in real-world requirements engineering contexts, the ambiguity threshold can be set by the users to help find the optimal tolerance to ambiguity. The optimal threshold needs to be found experimentally, and moreover might vary in the course of the requirements elicitation and analysis. For example, an analyst might want to focus initially on the very bad ambiguities, those that are almost guaranteed to cause trouble, and thus set a high threshold. As the analysis progresses and the most urgent cases of nocuous ambiguities are addressed (by clarifying, rewriting, or explicitly acknowledging them), the threshold can be progressively lowered, so that at each stage only a limited number of cases are shown (thus avoiding being overloaded with too many alarms).

In this scenario, which particular threshold was used in our experiments is not particularly sensitive, since in practice a large range of values will be used – from the very lax for an initial brainstorming , to the most exacting for a high-assurance system. Additionally, as shown in Section 7.1, the performances of the techniques stays in the range of 75%-80% accuracy as the threshold vary, so the applicability of the technique in real-life requirements engineering is not hindered by the particular threshold setting.

Applicability to other requirements engineering practices. We focused in this work on requirements written in free-form natural language. Therefore there is a threat that the results cannot be generalized to other contexts, such as when requirements are expressed as use cases, scenarios, or in semi-formalized or formalized languages. The scope of applicability of our results are in fact limited to those cases where a reasonably large part of the requirements are conveyed in natural language form, either as part of a large document, or as snippets in a structured document. The latter covers, among others, most scenarios and other template-based approaches. We do not claim applicability to other techniques used in current practice (e.g., goal models, SCR tables, fully formal specifications).

An interesting issue arises with regard to controlled natural language, which is mandated in several industries as the de-facto standard for writing requirements. In some cases, these languages exclude pronouns entirely, and hence our results (which pertain to pronominal anaphoric references) are not applicable. In other cases, pronouns are allowed, but the verbs or other connectives are restricted. We speculate that, for such controlled languages, results from the machine-learning algorithms would be even better (i.e. more precise) than what we have obtained from the generic corpus. Whether this would lead to a more precise approximation of the stakeholders' appreciation of ambiguity on the controlled language, is an open question.

9. Related Work

Ambiguity in Requirements Engineering. Ambiguity is a pervasive phenomenon in natural language, and thus in natural language requirements documents. The general phenomenon has been studied in philosophy and logic, and used in poetry in Homeric times. We will thus restrict ourselves to the specific treatment of ambiguity (and in particular, pronominal anaphora) in relation to requirements engineering. Early studies [16] have discussed '*communication errors*' in relation to general quality of requirements, but only in 1992 the particular case of pronoun references is specifically addressed [45]. In the latter, anaphora is used as an example of '*language ambiguities*', a very generic category which includes all cases where confusion or misunderstanding can derive from the specific wording used in a requirement. Four ambiguity categories that typically occur in requirements documents had been identified by Berry and his colleagues [3], which include lexical ambiguity, syntactic ambiguity, semantic ambiguity and pragmatic ambiguity. The second category, syntactic ambiguity, is the one more directly related to the case of pronominal anaphora that we have studied in the present work, although the *resolution* of the ambiguity is often driven by a combination of lexical, syntactic, semantic and pragmatic means (as discussed in Section 4.2.1). A more detailed discussion about the ambiguity types contained in each category, together with relevant examples is given in [3]. Furthermore, Gervasi and Zowghi [17] investigated the nature of ambiguity in requirements specifications and provided deeper analysis on the causes and effects of different types of ambiguity (focusing on lexical, syntactic, and semantic ambiguity) in the system development process in order to help better understand the role of ambiguity in RE practices. In that work, Gervasi and Zowghi suggest a role for the linguistic feature of *markedness* as a predictor of whether any ambiguity is intentional on

the part of the writer, or not. Compared to our approach, markedness could constitute a further heuristics to detect the potential nocuity of a specific instance of ambiguity.

Moving from more speculative works into the application-oriented ones, several studies have attempted to identify ambiguity in requirements for the purpose of improving the quality of NL requirements documents. A number of tools have been developed specifically to detect, measure, or reduce possible ambiguities in text. Fuchs and Schwitter [14] present a restricted NL, called Attempt Controlled English (ACE), to translate specifications into sentences in first-order logic (and, in successive works, extended to target a number of precisely defined semantics, such as PQL or RuleML) in order to reduce the ambiguity in requirement specifications. By design, requirements written in ACE have a unique interpretation, given by a set of rules. In particular, pronominal anaphora is allowed only in specific contexts (whereas cataphora is not allowed at all), and the ACE Interpretation Rules state that each pronoun has to be interpreted as referring to the most specific and most recent accessible noun phrase. In prescribing a fixed interpretation, ACE essentially dispel the ambiguity, but only with reference with the given standardized interpretation. However, a stakeholder who is not aware of the subtleties of ACE might still read the controlled language as if it was unrestricted natural language, and assume a wrong interpretation. In this sense, our technique is complementary to the usage of controlled languages of this sort, in that we point out ambiguities that could be misinterpreted, whereas ACE focuses on prescribing which interpretation is correct – without considering the risk of misinterpretation by a reader.

Mich and Garigliano [32] investigate the use of a set of ambiguity indices for the measurement in syntactic and semantic ambiguity, which is implemented using an NLP system called NL-OOPS [33], based on the generic natural language processing system LOLITA [37]. NL-OOPS is aimed mostly at deriving object-oriented models of software from natural language requirements, and they consider ambiguity mostly as an obstacle to the derivation of a single model. In details, the ambiguity measures they use are based on three factors: *polisemy*, that is the number of different senses a word has (in a reference lexical resource), the number of different part-of-speech roles that a word could occupy, and how difficult it was for the LOLITA system to compute a parse tree for the sentence (essentially, measuring the uncertainty of the parse tree); based on these three factors, other derived measures are defined. This approach is different than our own, in that in the case of [32], a definition of what constitute ambiguity is given *a priori* by defining the metrics apodictically (and hence, the appropriateness of these metrics to the stakeholders' intuition has to be proven *a posteriori*), whereas in our technique, what constitutes a nocuous ambiguity is learned via ML from the stakeholders' responses. Similarly, Boyd et al. [6] describe a controlled natural language to help reduce the degree of lexical ambiguity of requirements specifications. By substituting synonyms or hyponyms with corresponding terms, and thus obtaining a reduced vocabulary. This approach helps with pronominal anaphora in that it reduces the chances for multiple references, but the issue is not discussed in detail in their work.

Kamsties and his colleagues [23] describe a pattern-driven inspection technique to detect ambiguities in NL requirements; the technique however is essentially human-driven, and thus can draw on the knowledge of an expert inspector. When applied by a skilled professional, the technique can be effective, but it has higher costs and lower repeatability then the tool-supported

technique we proposed. We regard as a particularly important point that with our technique, nocuity detection can be performed repeatedly and with predictable effectiveness at different times during the requirements analysis process. Berry et al. [5] presented a natural language requirements specification model to address expressiveness and structural consistency problems of natural language requirements by examining the ambiguities manifested at the lexical, syntactic, structure, and semantic level. Kiyavitskaya et al. [28] proposed a two-step approach in which a set of lexical and syntactic ambiguity measures are firstly applied to ambiguity identification, and then a tool reports about what specifically is potentially ambiguous in each sentence.

Another strand of research has considered ambiguity among the quality features of requirements documents, and has set to develop tools to measure quality attributes in general. Most of these approaches define a quality model (QM) composed of a set of quality metrics (e.g., vagueness, subjectivity, optionality, weakness, etc.), and develop analysis techniques based on linguistic approaches to detect the defects (we are interested here in those related to the inherent ambiguity in the requirements). For example, QuARS (Quality Analyzer of Requirements Specification) [12] is a linguistic language tool based on a quality model for NL requirements specifications. It aims to detect lexical, syntactic, structural, and semantic defects including ambiguities. In QuARS, certain terms or syntactic structures are considered "dangerous" by themselves; for example, use of certain adverbs (e.g., '*sufficiently*') or syntactic structures (e.g., coordination) are marked as potentially nocuous. The main obstacle to applying this approach in practice is the rather high number of false positives; in fact, there is no analysis of which among the potentially dangerous constructs are likely to really cause interpretation problems to the stakeholders. Wilson et al. [55] developed a QM tool, ARM (Automated Requirement Measurement), to identify potential problems, such as ambiguity, inaccuracy, and inconsistency, in natural language specification statements. Fantechi et al. [13] proposed a linguistic approach to detect the defects, such as vagueness, subjectivity, weakness, which are caused by ambiguity at the sentence level in functional requirements of textual (NL) user cases. Kaiya and Saeki [24] made use of the semantic relationships between concepts on a domain ontology to check if the ambiguity property of a require item. Achour [1] discussed the problem of requirements vague and unverifiable caused by ambiguity of words and phrases. All these techniques share the same approach as QuARS: the final validation with human stakeholders is used to assess the effectiveness of the technique, but not to inform the actual nocuity classification algorithm. In this sense, the technique we have proposed is novel compared to these previous experiences.

Finally, there is a body of work that reported the effect of ambiguity problem on non-textual artifacts. The problem faced is significantly different from the one we discussed in this paper, in that the amount of text used as labels in most of these approaches is too limited to allow any substantial scope for ambiguity. Still, they share our overall goal, and are worth mentioning for completeness. For example, van Rossum [51] addressed the effects of technology implementation on the ambiguity of organization goals, and Futrelle [15] discussed the diagram ambiguity problem including the ambiguities inherent in the text within the diagram. Harter et al. [21] discussed the side effect of requirements ambiguity with respect to both development effort and cycle time that

are linked to software quality. More generally, Sussman and Guinan [48] conducted a study on the role of task ambiguity in the context of software development.

Anaphora Resolution. Anaphora resolution research has been attracted intensive attentions from NLP community in the last few decades, and approaches to the anaphora resolution problem also vary. Earlier research efforts had focused on traditional heuristic-based approaches that explored various types of heuristics, typically involving the use of lexical, syntactic and semantic information [31, 35], computational theories of discourse such as focusing [20] and centering algorithm [50, 54], and statistics/corpus approach based on co-occurrence patterns [10, 41]. In recent, supervised machine learning approaches have been widely explored in reference resolution and achieved considerable success. A variety of ML algorithms, such as decision tree [2, 46, 38, 47], Latent Sematnic Analysis (LSA) [29] and Support Vector Machine (SVM) [22], are investigated to determine the referent relationship between an anaphor and its antecedent candidate from the properties of the pair by using various types of features including string-matching features, syntactic features, grammatical features, semantic features, and discourse-based features. A broad overview of anaphora or NP coreference research can be found in books [36], tutorials [43], or overview papers [39]. In addition, a number of research projects have applied the anaphora resolution techniques in various kinds of applications such as machine translation [44], and information extraction like named entity identification [8, 27].

Similarly to other anaphora resolution approaches, we have explored a number of antecedent preferences to discover the preference for the candidates to the anaphor. However, our tool differs from the related work in that we have attempted to determine the degree to which an ambiguity is likely to be misinterpreted (i.e. whether it is nocuous relative to a particular ambiguity threshold), rather than attempting to undertake the anaphora resolution by applying disambiguation techniques to select the most like candidate as the antecedent of the anaphor.

10. Conclusions and Future Work

Since many requirements documents continue to be written in natural language, we need ways to deal with the ambiguity inherent in natural language. Our overall research goal is to develop techniques to detect potential nocuous ambiguity in requirements in order to minimize its effects.

In this paper, we extended our previous work on anaphoric ambiguity by introducing an overall conceptual architecture of an automated system to identify potentially nocuous ambiguities in requirements documents that contain anaphoric ambiguity. Given a natural language requirements document, sentences that contain anaphoric ambiguities are first selected automatically from the text, and then a list of NP antecedent candidates are extracted from each anaphoric ambiguity instance. Possible coreference relationships among the NP candidates are identified using an NP coreference resolution engine.

To construct a machine learning based antecedent classifier, a set of anaphoric ambiguities were extracted from a range of requirements documents, and associated human judgments on their interpretations were collected. The antecedent classifier was trained based on a set of antecedent preference heuristics and collected human judgments, which was used to predict the antecedent

preference of noun phrase antecedent candidates. The antecedent information was then used to identify nocuous ambiguity. Our experimental results showed that our approach achieves high recall with a consistent improvement on baseline precision subject to appropriate ambiguity thresholds, allowing us to highlight realistic and potentially problematic ambiguities in actual requirements documents.

Although based on significant technical development and substantive empirical studies, we believe that the application of our approach is actually lightweight and usable in that it allows requirements analysts to experiment and iterate to identify potential nocuous ambiguity in requirements, depending on their chosen analysis sensitivity threshold.

Nevertheless, there are a number of areas for improvement. For prediction accuracy of nocuous ambiguity, a larger dataset is required in order to obtain more training instances for the construction of the antecedent classifier. One of the problems for our current classifier is the shortage of *positive* and *questionable* instances compared with *negative* instances. We expect that more positive/questionable instances would enhance the accuracy of the classifier. More research is also needed to exploit additional antecedent preferences that account for more aspects of anaphora.

In earlier work [56], we presented a methodology for the identification of nocuous ambiguity, which described three basic ideas underpinning our model of ambiguity: collection of human judgments, the heuristics used to model human interpretations, and a machine learning module to train the heuristics. This methodology has now been used effectively to handle coordination ambiguity [9, 54, 57] and anaphoric ambiguity [58]. We intend to apply the methodology to a wider range of ambiguity types, such as scope and preposition ambiguity, which also contain rich syntactic and semantic information that can be modeled by various heuristics.

Currently, our automated tools are still prototypes. It will be interesting to see if they are actually useful in practice, when applied in real-world requirements engineering environments. A number of general research questions need to be investigated in this context. How often does a requirements engineering problem arise because of an ambiguity problem? And, how many errors reported in actual applications be tracked back to an ambiguity such as nocuous anaphoric ambiguity?

We envision that our automated support for ambiguity analysis will fit into one of a number of requirements management environments, in which requirements authors are able to invoke our analysis tool in much the same way as writers invoke spell checkers. We are currently investigating the development of this capability within a well known commercial tool.

Acknowledgments. This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) as part of the MaTREx project (EP/F068859/1), and by the Science Foundation Ireland (SFI grant 03/CE2/I303_1). We are grateful to our research partners at Lancaster University for their input, and to Ian Alexander for his practical insights and guidance. Moreover, we also wish to acknowledge the anonymous reviewers' insightful comments and suggestions.

References

1. Achour CB, Rolland C, Souveyet C, Maiden NAM (1999) Guiding Use Case Authoring: Results of an Empirical Study. In: Proceedings of 7th IEEE International Requirements Engineering Conference (RE'99) pp 36-43
2. Aone C, Bennet SW (1996) Applying machine learning to anaphora resolution. In: Connectionist, Statistical and symbolic approaches to learning for natural language processing. pp 302-314
3. Berry DM, Kamsties E, Krieger MM (2003) From contract drafting to software specification: Linguistic sources of ambiguity.
4. Berry D, Kamsties E (2005) The syntactically dangerous all and plural in specifications IEEE Softw 22:55-57
5. Berry D, Bucchiarone A, Gnesi S, Lami G, Trentanni G (2006) A new quality model for natural language requirements specifications. In: Proceedings of the international workshop on requirements engineering: foundation of software quality (REFSQ)
6. Boyd S, Zowghi D, Farroukh A (2005) Measuring the expressiveness of a constrained natural language: An empirical study. In: Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE'05), Washington, DC, pp 339-352
7. Brennan SE, Friedman MW, Pollard C (1987) A centering approach to pronouns. In: Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics (ACL), pp 155–162
8. Castaño J, Zhang J, Pustejovsky H (2002) Anaphora Resolution in Biomedical Literature. In: Proceedings of International Symposium on Reference Resolution
9. Chantree F, Nuseibeh B, de Roeck A, Willis A (2006) Identifying Nocuous Ambiguities in Natural Language Requirements. In: Proceedings of 14th IEEE International Requirements Engineering Conference (RE'06), Minneapolis, USA, pp 59-68
10. Dagan I, Itai A (1990) Automatic processing of large corpora for the resolution of anaphora references. In: Proceedings of the 13th International Conference on Computational Linguistics (COLING'90) pp 1-3
11. Denber M (1998) Automatic resolution of anaphora in English. Eastman Kodak Co., Technical Report
12. Fabbrini F, Fusani M, Gnesi S, Lami G (2001) The linguistic approach to the natural language requirements, quality: benefits of the use of an automatic tool. In: Proceedings of the twenty sixth annual IEEE computer society—NASA GSFC software engineering workshop, pp 97–105
13. Fantechi A, Gnesi S, Lami G, Maccari A (2003) Applications of Linguistic Techniques for Use Case Analysis. Requir Eng J 8 (9):161-170
14. Fuchs NE, Schwitter R (1995) Specifying logic programs in controlled natural language. In: Proceedings of the Workshop on Computational Logic for Natural Language Processing, pp 3–5

15. Futrelle RP (1999) Ambiguity in visual language theory and its role in diagram parsing. In: Proceedings of the IEEE symposium on visual languages (VL'99), IEEE Computer Society, p. 172
16. Gause DC, Weinberg GM (1989) Exploring Requirements: Quality before Design. Dorset House, New York
17. Gervasi V, Zowghi D (2010) On the Role of Ambiguity in RE In: Proceedings of the 16th International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ), pp 248-254
18. Gnesi S, Lami G, Trentanni G, Fabbrini F, Fusani M (2005) An Automatic Tool for the Analysis of Natural Language Requirements. Int J of Comput Syst Sci & Eng (IJCSSE) 2 (1):53-62
19. Goldin L, Berry DM (1997) AbstFinder, A Prototype Natural Language Text Abstraction Finder for Use in Requirements Elicitation. Autom Softw Eng 4 (4):375-412
20. Grosz BJ, Joshi AK, Weinstein S (1995) Centering: A framework for modeling the local coherence of discourse. Computational Linguistics 21 (2):203–226
21. Harter DE, Krishnan MS, Slaughter SA (1998) The life cycle effects of software process improvement: a longitudinal analysis. In: Proceedings of the international conference on information systems, association for information systems, pp 346–351
22. Iida R, Inui K, Matsumoto Y (2005) Anaphora resolution by antecedent identification followed by anaphoricity determination. ACM Trans on Asian Lang Inf Process (TALIP) 4 (4):417 - 434
23. Kamsties E, Berry D, Paech B (2001) Detecting ambiguities in requirements documents using inspections. In: Proceedings of the First Workshop on Inspection in Software Engineering (WISE'01), pp 68-80
24. Kaiya H, Saeki M (2006) Using Domain Ontology as Domain Knowledge for Requirements Elicitation. In: Proceedings of 14th IEEE International Requirements Engineering Conference (RE'06) pp 186-195
25. Keren G (1992) Improving decisions and judgments: The desirable versus the feasible. In: Wright G, Bolger F (eds) Expertise and decision support. Plenum Press, pp 25-46
26. Kilgarriff A, Rychly P, Smrz P, Tugwell D (2004) The Sketch Engine. In: Proceedings of the Eleventh European Association for Lexicography (EURALEX), pp 105–116
27. Kim J, Jong CP (2004) BioAR: Anaphora Resolution for Relating Protein Names to Proteome Database Entrie. In: Proceedings of ACL Workshop on Reference Resolution and its Applications pp 79-86
28. Kiyavitskaya N, Zeni N, Mich L, Berry DM (2008) Requirements for tools for ambiguity identification and measurement in natural language requirements specifications. Requir Eng J 13:207–240
29. Klebanov B, Wiemer-Hastings PM (2002) Using LSA for Pronominal Anaphora Resolution. In: Proceedings of the Third International Conference of Computational Linguistics and Intelligent Text Processing (CICLing 2002), Mexico City, Mexico, pp 197-199

30. Kotonya G, Sommerville I (1998) Requirements Engineering Processes and Techniques. John Wiley & Sons
31. Lappin S, Leass H (1994) An Algorithm for Pronominal Anaphora Resolution. Comput Linguist:535-561
32. Mich L, Garigliano R (2000) Ambiguity measures in requirement engineering. In: Proceedings of international conference on software—theory and practice (ICS2000), pp 39–48
33. Mich L, Garigliano R (2002) NL-OOPS: a requirements analysis tool based on natural language processing. In: Proceedings of third international conference on data mining, pp 321–330.
34. Mich L, Franch M, Inverardi PN (2004) Market research for requirements analysis using linguistic tools. Requir Eng J 9:40–56
35. Mitkov R (1998) Robust pronoun resolution with limited knowledge. In: Proceedings of the 18th International Conference on Computational Linguistics (COLING'98)/ACL'98 Montreal Canada pp 869-875
36. Mitkov R (2002) Anaphora Resolution. Longman.
37. Morgan R, Garigliano R, Callaghan P, Poria S, Smith M, Urbanowicz A, Collingham R, Costantino M, Cooper C (1995) Description of the LOLITA system as used in MUC-6. In: Proceedings of the sixth message understanding conference (MUC-6. (1995))
38. Ng V, Cardie C (2002) Improving Machine Learning Approaches to Coreference Resolution. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pp 104-111
39. Ng V (2010) Supervised Noun Phrase Coreference Research: The First Fifteen Years. In: Proceedings of the 48nd Annual Meeting of the Association for Computational Linguistics (ACL-2010), pp 1396–1411
40. Oliver DE, Bhalotia G, Schwartz AS, Altman RB, Hearst MA (2004) Tools for loading Medline into a local relational database. BMC Bioinforma 5:146
41. Paul M, Yamamoto K, Sumita E (1999) Corpus-based anaphora resolution towards antecedent preference. In: Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics, Workshop "Coreference and It's Applications", pp 47-52
42. Ponzetto SP, Poesio M (2009) State-of-the-art NLP approaches to coreference resolution: Theory and practical recipes. In: Tutorial Abstracts of ACL-IJCNLP 2009, pp 6.
43. Poesio M, Artstein R (2008) Introduction to the Special Issue on Ambiguity and Semantic Judgements. Res on Lang & Comput 6:241-245
44. Saggion H, Carvalho A (1994) Anaphora resolution in a machine translation system. In: Proceedings of the International Conference on Machine Translation, pp 1-14
45. Schneider GM, Martin J, Tsai WT (1992) An Experimental Study of Fault Detection in User Requirements Documents. ACM Trans on Softw Eng and Methodol 1 (2):188-204
46. Soon WM, Ng HT, Lim DCY (2001) A machine learning approach to coreference resolution of noun phrases. Computational Linguistics 27:521–544

47. Strube M, Muller C (2003) A machine learning approach to pronoun resolution in spoken dialogue. In: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL), pp 168–175
48. Sussman SW, Guinan PJ (1999) Antidotes for high complexity and ambiguity in software development. Inf Manage 36:23–35
49. Tetreault JR (2001) A corpus-based evaluation of centering and pronoun resolution. Computational Linguistics 27 (4):507-520.
50. Tsuruoka Y, Tateishi Y, Kim J, Ohta T, McNaught J, Ananiadou S (2005) Developing a Robust Part-of-Speech Tagger for Biomedical Text. In: Advances in Informatics. pp 382-392
51. van Rossum W (1997) The implementation of technologies in intensive care units: ambiguity, uncertainty and organizational reactions. Technical Report Research Report 97B51, Research Institute SOM (Systems, Organisations and Management), University of Groningen, Groningen, The Netherlands, http://irs.ub.rug.nl/ppn/165660821 or http://ideas.repec.org/p/dgr/rugsom/97b51.html#download
52. Wasow T, Perfors A, Beaver D (2003) The Puzzle of Ambiguity. In: Orgun O, Sells P (eds) Morphology and the Web of Grammar: Essays in Menory of Steven G. Lapointe.
53. Walker M, Joshi A, Prince E (1998) Centering Theory in Discourse. Oxford University Press.
54. Willis A, Chantree F, de Roeck A (2008) Automatic Identification of Nocuous Ambiguity. Res on Lang & Comput 6 (3-4):1-23
55. Wilson WM, Rosenberg LH, Hyatt LE (1997) Automated analysis of requirement specifications. In: Proceedings of the Nineteenth International Conference on Software Engineering (ICSE), pp 161–171
56. Yang H, de Roeck A, Willis A, Nuseibeh B (2010) A Methodology for Automatic Identification of Nocuous Ambiguity. In: The 23th International Conference on Computational Linguistics (Coling'10), pp 1218-1226
57. Yang H, Willis A, de Roeck A, Nuseibeh B (2010) Automatic Detection of Nocuous Coordination Ambiguities in Natural Language Requirements. In: The 25th IEEE/ACM International Conference on Automated Software Engineering (ASE'2010), pp 53-62
58. Yang H, de Roeck A, Gervasi V., Willis A, Nuseibeh B (2010) Extending Nocuous Ambiguity Analysis for Anaphora in Natural Language Requirements. In: Proceedings of 18th IEEE International Requirements Engineering Conference (RE'10) pp 25-34