Speculative Requirements:
Automatic Detection of Uncertainty in Natural Language Requirements

Hui Yang[1] Anne De Roeck[1] Vincenzo Gervasi[2] Alistair Willis[1] Bashar Nuseibeh[1, 3]

[1] Department of Computing, The Open University, UK
[2] Department of Computer Science, University of Pisa, Italy
[3] Lero-The Irish Software Engineering Research Centre, University of Limerick, Ireland
e-mail: {h.yang, a.deroeck, a.g.willis, b.nuseibeh}@open.ac.uk; gervasi@di.unipi.it

Abstract: Stakeholders frequently use speculative language when they need to convey their requirements with some degree of uncertainty. Due to the intrinsic vagueness of speculative language, speculative requirements risk being misunderstood, and related uncertainty overlooked, and may benefit from careful treatment in the requirements engineering process. In this paper, we present a linguistically-oriented approach to automatic detection of uncertainty in natural language (NL) requirements. Our approach comprises two stages. First we identify speculative sentences by applying a machine learning algorithm called Conditional Random Fields (CRFs) to identify uncertainty cues. The algorithm exploits a rich set of lexical and syntactic features extracted from requirements sentences. Second, we try to determine the scope of uncertainty. We use a rule-based approach that draws on a set of hand-crafted linguistic heuristics to determine the uncertainty scope with the help of dependency structures present in the sentence parse tree. We report on a series of experiments we conducted to evaluate the performance and usefulness of our system.

Keywords: Uncertainty, natural language requirements, speculative requirements, uncertainty cues, machine learning, uncertainty scopes, rule-based approach

I. INTRODUCTION

Many natural language (NL) requirements are stated in a tentative or speculative manner. In some requirements *uncertainty* may have been included deliberately, to avoid committing to factual statements about which the author is unsure, or included semi-intentionally, for example when transcribing an interview, as the expression of an inherent uncertainty in the requirements. It is often the case that such statements contain linguistic cues that appear in the requirements text. Consider the following examples:

E1. *The vending machine will offer mineral water.*
E2. *It is possible that the vending machine might offer mineral water.*

(E1) is framed as a *factual* statement: the action is certain (mineral water will be provided), and the author of (E1) is certain about it. In contrast, (E2) is in the form of a *speculative* statement: it exhibits multiple uncertainty cues (e.g., *possible, might*) that qualify the author's confidence in the truth of a proposition [6]. In (E2) the author is uncertain about whether the vending machine provides mineral water or not. Moreover, notice that the distinction between factual and speculative statements is not the same as between indicative (state of the domain) and optative (desired behaviour of the system), as defined by Jackson [8]. In fact, one could have speculative indicative statements (i.e. uncertainty in the state of the domain) and speculative optative statements (i.e. uncertainty about the desired behaviour), in addition to the factual versions which the literature commonly considers.

Uncertainty in requirements documents has several undesirable effects. It can lead to system behaviour that does not meet users' expectations, if no proper analysis of the root causes of the uncertainty is performed, and alternatives are not considered. It can also lead to untestable requirements and makes it difficult to plan and estimate development costs. It can cause developers to substitute their own preferences and expectations for the speculative requirements. In short, speculative requirements that survive till the implementation phase are potentially harmful.

However, uncertainty can have a positive role to play in the early stages of requirements analysis. If speculative requirements are identified, they may act as indicators of areas where further elicitation is needed. If more detailed conditions, risks, and exceptional behaviours are uncovered, they may be encoded in the requirements. If on the other hand there is a real lack of specific preference, speculative requirements may leave a larger design space for developers to explore solutions. Thus there can be a positive side to identifying speculative requirements.

We suggest that uncertainty cues should be identified and flagged at an early stage, possibly as soon as requirements are written down. Moreover, regardless of whether speculative language is seen as an undesirable attribute of requirements [3] or whether it is seen as having a positive role in the elicitation process, we suggest that correctly identifying and classifying instances of uncertainty in requirements is important. Our research is motivated by the need to investigate the use of speculative language in natural language requirements. Our practical goal is to support requirements analysts in making in-depth analyses of reported cases by providing automated tools that highlight linguistic expressions in NL requirements that are recognised as speculative statements.

In this paper we report on our work of identifying expressions of uncertainty in NL requirements by means of a linguistically-oriented automated tool. We have evaluated our automated system on an uncertainty dataset of 11 full-text requirements documents in which uncertainty cues and their scopes have been manually annotated according to estab-

lished annotation guidelines[1]. We report the preliminary results of our system's performance in terms of speculative sentence identification and uncertainty scope resolution.

The rest of the paper is structured as follows. In Section II, we discuss the property of uncertainty, and analyse different types of uncertainty that occur in NL requirements documents. Section III provides a detailed description of our technique for identifying speculative sentences. Section IV presents our rule-based approach to determining the scope of uncertainty cues. Our experimental results are reported in Section V. Section VI addresses potential threats to validity of our findings. Section VII discusses related work, and conclusions and future work are presented in Section VIII.

II. Uncertainty in Natural Language Requirements

A. Uncertainty detection

Uncertainty, or more generally, *hedging* or *speculation*, is a language component that is often used to express tentativeness, skepticism, or doubt when authors are not completely certain of their claims/statements. Speculative language is a communicative strategy for weakening the force of a statement. It is usually triggered by particular words (e.g., *possible*, *might*, *likely*) or phrases (e.g., *not sure*, *whether or not*), called *uncertainty cues*, which weaken some clauses or propositions. In requirements, stakeholders often have a need to qualify the degree of confidence that they have in certain sentences. Attempts to deprive them of natural ways of expressing such qualifications (e.g., by forbidding the use of certain words), will be impractical during requirements elicitation. Indeed we believe there is little point in trying to enforce absolute certainty if and when, in reality, stakeholders can offer only tentative speculations.

In any case, the occurrence of such cues is insufficient as a sure indicator of uncertainty. For example, both requirements (E3) and (E4) below contain the cue '*appear*', yet (E3) is a factual statement with no uncertainty. Therefore, the correct identification of uncertainty needs to be sensitive to the linguistic *context* – i.e. the surrounding words – in which the cues occur.

E3. *The system shall allow an operator to assign colours for countries that appear on the tactical display.*

E4. *It appears that Insulin should be delivered in circumstances where the level is likely to go outside this range.*

Similarly, (E5) below is a genuinely speculative statement: there is uncertainty about whether the supplier will submit the overview or not, although there is a clear indication of preference. However, in (E6) there is nothing speculative about the fact that modifications should be typeset in italics, as being '*suggested*' is a factual property of a modification. In other words, uncertainty has a *scope* that may relate to a situation captured in the requirement statement.

[1] http://www.inf.u-szeged.hu/rgai/project/nlp/bioscope/Annotation%20guidelines2.1.pdf

E5. *It is strongly suggested that the supplier submits an overview of the alternative, the stages involved for each, and the releases.*

E6. *Modifications suggested during the review of the DRAFT version of the SPS will be added in italics to preserve the original notes.*

Our method for detecting uncertainty reflects this analysis and operates in two stages: the identification of speculative sentences and the determination of uncertainty scope.

The first stage - *speculative sentence identification* - labels each sentence in a requirements document as either speculative or non-speculative. As shown in the examples above, uncertainty cues cannot be determined just by simple keyword matching approaches, but need to recognise the possible multiple word senses involved in a word (such as *appear*), and the characteristics of the surrounding context.

In this paper, we used a machine learning approach, first to identify a number of linguistic features typical of speculative sentences, and then to applying a Conditional Random Fields (CRFs) algorithm [10] in order to learn models that classify whether or not a given instance of an uncertainty cue is used speculatively.

The second stage is *uncertainty scope resolution* which attempts to identify which fragments of a sentence are affected by the speculative expressions; that is, the scope of uncertainty cues. Particularly in long, complex sentences, only parts of the sentence are speculative as in (E7)[2].

E7. *The text interface will provide equivalent functionality, but will [likely require additional server requests to perform the same task].*

In this case, the scope of the speculation keyword '*likely*' only spans the second coordinated clause of the sentence, '*likely require additional server requests to perform the same task*', while the first clause, '*The text interface will provide equivalent functionality*', conveys factual information. Therefore, it is necessary to discriminate between speculative and non-speculative fragments at the sentence level in order to avoid information loss.

We have developed an automated rule-based approach to identify the linguistic scope of speculative cues. We devised a small set of hand-crafted rules, which rely heavily on syntactic structure information as well as on various additional features related to specific syntactic constructions of the uncertainty cue in focus. These rules are applied to the parse trees of sentences containing speculative clues, extracted by the Stanford Parser[3].

B. Uncertainty cue categories

Uncertainty in natural language may be realised through various linguistic cues and is marked by a variety of syntactic constructions. We drew on Hyland's study of the lexical

[2] Our examples are adapted from our collection of requirements documents. We underline uncertainty cues and highlight uncertainty scope with square brackets.

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

surface realisation of uncertainty [6], and analysed syntactic structures expressing uncertainty in the BioScope Corpus [17]. Typical uncertainty cues fall within the following six categories, which are classified based on the generalised part-of-speech (POS) tag of the cue. Some examples of these categories are given below.

- **Auxiliaries**

- *may, might, can, would, should, could, etc.*

E8. *Please consider carefully before disclosing any personal information that [might be accessible to others].*
E9. *[The user authorizations should be done through the LDAP mechanism].*

- **Epistemic verbs**

- *suggest, presume, suppose, seem, appear, indicate, etc.*

E10. *This [assumes that service center computers have access to the Internet].*
E11. *[There appears to be a need to enter in all the information in one system and have it update all other systems].*

- **Epistemic adjectives**

- probable, possible, likely, unlikely, unsure, not sure, etc.

E12. *Two [possible display page formats could be used].*
E13. *[The procedure for aligning calibration data is still unclear].*

- **Epistemic adverbs**

- *probably, possibly, presumably, perhaps, potentially, etc.*

E14. *[Perhaps this is a phenomenon restricted to programmers], as opposed to literature authors.*
E15. *A system consisting solely of software and [possibly the computer equipment on which the software operated.*

- **Epistemic nouns**

- *possibility, probability, hypothesis, suggestion, etc.*

E16. *There is the [possibility that existing products, such as a process manager, will be used].*
E17. *The parameter values are stored to provide the [possibility to have different characteristics of the outside temperature].*

- **Conjunctions**

- *or, and/or, either … or, whether … or, whether or not, etc.*

E18. *Web sites hosting software are maintained [either by the LMI department or by another state agency].*
E19. *If the owner has made a counter offer, the ROW Chief decides [whether or not to accept the counter offer].*

Note that the scope of an uncertainty cue varies quite significantly, and is related to syntactic patterns associated with the keyword cue.

C. Uncertainty and ambiguity

In previous work [19], we have argued that ambiguity is common in NL requirements. In cases where different stakeholders interpret the same text differently (described as '*nocuous ambiguity*'), then systems risk being incorrectly implemented. If such ambiguity can be detected in a requirements document, then the requirements writer should be *notified* with a view to clarifying the text and removing (so far as is possible) the ambiguity.

Uncertainty differs from ambiguity both in root cause and in remedial actions. Ambiguity arises (in part) as a result of the *form* of the requirement. The writer may be quite certain of the desired outcome, and may not even realise that the writing is ambiguous, but the particular choice of words and expression might lead other stakeholders to interpret the text incorrectly.

Uncertainty is a property of the *contents* of a requirement, and is often indicated through the writer's use of speculative language. In fact, speculative language denotes an *explicit* admission by the stakeholder that the information conveyed is not reliable (unlike ambiguity, where the writer might be unaware of this). We could thus consider speculative language to be an explicit annotation of risk - given in literary terms instead of as a numeric assessment.

As a consequence, remedial actions are very different in the two cases. For an ambiguous requirement, an analyst ought to ask which of the possible interpretations was intended ('*Did the writer mean A or B?*'), and ensure that the requirement is rewritten in a less ambiguous way (probably in consultation with the original writer). For speculative requirements, the analyst should focus on why the stakeholder is uncertain, what is causing the uncertainty, what are the risk factors, what other scenarios are possible, whether there are exceptional cases to be considered, and how the system should handle them. At times, this might result in rewriting a speculative requirement ('*Maybe A*') as a series of non-speculative, conditional requirements ('*If Z then A else B*' or '*A, unless Z: in which case B*').

In fact, it is possible for the two phenomena to be conflated in the same requirement. A truly speculative requirement can be framed in ambiguous language, or ambiguity could be consciously used as one of the many linguistic devices used to convey uncertainty. This phenomenon is sometimes seen in those cases for which the consequences of both ambiguity and uncertainty tend to the same result. In other words, if a stakeholder used ambiguous language to convey uncertainty, then misclassifying that speculative requirement as ambiguous would still prompt the analyst to look more closely at it, and the classification error could be promptly corrected by human review. Hence, although instances of ambiguity and uncertainty are to be treated differently, where they interact, our proposed remedial actions will converge on a common result of a more complete requirements document which is less susceptible to misinterpretation.

D. Uncertainty dataset

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. Although both our previous work on ambiguity [19] and the

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

present one on uncertainty make use of the same set of requirements documents, the collected sample instances used for the system development are different.

In the work presented in this paper, we manually annotated uncertainty cues and their corresponding scopes. The statistics of this uncertainty dataset are given in Table I. The speculative sentences contain different categories of uncertainty cues described above, and the distribution of individual cue categories is shown in the Figure 1. The top 10 frequently occurring uncertainty cues in these requirements documents are *may* (365), *should* (130), *possible* (83), *or* (67), *whether* (57), *would* (47), *indicate that* (39), *could* (20), *might* (18), *possibility* (18), respectively.

TABLE I. THE STATISTICS OF AN UNCERTAINTY DATASET ABOUT REQUIREMENTS DOCUMENTS

Documents	11
Sentences Speculative sentences	26, 829 914 (3.4%)
Cues Unique cues Cues with multiple words	1024 51 52 (4.4%)
Scope Scope with multiple cues	1003 21 (2.1%)

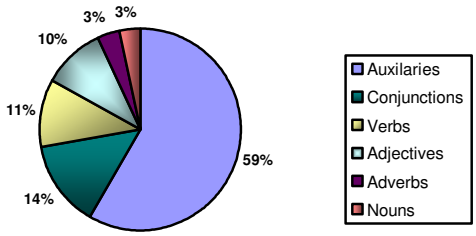

Figure 1. The distributions of different cue categories

III. DETECTION OF SPECULATIVE SENTENCES

This section describes how we identify the sentences in requirements documents that contain uncertain information. We approach it as a sentence classification problem, that is, any sentence containing at least one uncertainty cue is marked as candidate-speculative, to indicate that it may convey uncertainty, while any sentence with no cues is labelled clearly as non-speculative at this stage. As a result, the task of speculative sentence detection could be treated as a problem of recognizing whether uncertainty cues occur in contexts that allow them to convey uncertainty.

A. *Uncertainty cues*

Multi-word cues. As described in Section II.B, uncertainty is often triggered by a single word such as *might*, *possible* and *probably*. However, in some cases an uncertainty cue is expressed via a phrase that spans multiple tokens, e.g., *not sure*, *whether or not*. Some complex speculative keywords exhibit strong hedging strength only when they function at the whole phrase level, but not at the individual token level, such as '*remain to be determined*', '*cannot definitely confirm*'. The fact that most multi-word cues are very infrequent, some even occurring only once in our dataset, presents a problem because training a machine learning classifier requires syntactic pattern frequency. We address this by relying on a raw string matching approach to identify multi-word cues.

Weak cues. Not all uncertainty cues convey the same degree of speculation, and some cue keywords may not be used in a speculative context. For example, the modal auxiliary '*can*' is ambiguous between several meanings, e.g., ability (deontic), and possibility (epistemic). It is considered as an uncertainty cue only in its epistemic sense. This is also the case for some other modal auxiliaries (e.g., *can*, *could*, *should*, and *would*) which express uncertainty only when used in a particular sense, which may be identified from the surrounding context of the modal [2]. Such terms are called *weak cues*. Interestingly, some of the false positive cases in cue identification are due to word sense ambiguity in weak cues. For example, the verb '*appear*' exhibits different word senses in the example (E3) and (E4) as discussed in the introduction. The context around the uncertainty cue plays an important role in resolving weak cue instances and our approach is sensitive to possible relevant relations between a cue and surrounding tokens.

TABLE II. EXAMPLE OF UNCERTAINTY CUE TAGGING

WORD	LEMMA	PoS	CHUNK	H-L	H-PoS	DL	Co-oc	Cue
Tasks	*task*	*NNS*	*B-NP*	*have*	*VB*	*nsubj*	*O*	*O*
may	*may*	*MD*	*B-VP*	*have*	*VB*	*aux*	*O*	*B*
have	*have*	*VB*	*I-VP*	*ROOT*	*O*	*root*	*O*	*O*
many	*many*	*JJ*	*B-NP*	*milestone*	*NNS*	*amod*	*O*	*O*
milestones	*milestone*	*NNS*	*I-NP*	*have*	*VB*	*dobj*	*O*	*O*
which	*Which*	*WDT*	*B-NP*	*associate*	*VBN*	*nsubjpass*	*O*	*O*
may	*may*	*MD*	*B-VP*	*associate*	*VBN*	*aux*	*O*	*B*
or	*or*	*CC*	*O*	*associate*	*VBN*	*cc*	*O*	*I*
may	*may*	*MD*	*B-VP*	*associate*	*VBN*	*aux*	*O*	*I*
not	*not*	*RB*	*I-VP*	*associate*	*VBN*	*neg*	*O*	*I*
be	*be*	*VB*	*I-VP*	*associate*	*VBN*	*auxpass*	*O*	*O*
associated	*associate*	*VBN*	*I-VP*	*milestone*	*NNS*	*rcomd*	*O*	*O*
with	*with*	*IN*	*B-PP*	*associate*	*VBN*	*prep*	*O*	*O*
the	*the*	*DT*	*B-NP*	*task*	*NN*	*det*	*O*	*O*
task	*task*	*NN*	*I-NP*	*with*	*IN*	*pobj*	*O*	*O*
5.1	*5.1*	*CD*	*I-NP*	*task*	*NN*	*num*	*O*	*O*
.	*.*	*O*	*O*	*O*	*O*	*O*	*O*	*O*

B. *Identifying speculative keywords*

We formulate the problem of uncertainty cue detection as a token-level sequence labelling task. Table II shows a pre-processed sample sentence (E20) with the rich information per token. Each word token in a sentence is assigned one of the so-called ***BIO*** scheme tags: ***B*** (first word of a cue), ***I*** (inside a cue), ***O*** (outside, not in a cue) as shown in the last column in Table II.

E20. *Tasks may have many milestones which may or may not be associated with the task 5.1.*

We extracted a wide variety of linguistic features (See Table II), both syntactic and surface-oriented, which attempt to characterise the semantics of speculative keywords. The features used for the token classification were grouped into the following four categories:

- **Word-token Features.** This type of feature includes word lemma, Part-of-Speech (PoS) tag, and chunk tag of the word, which are obtained from the Genia Tagger[5].

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

- **Context Features.** The features for the lemma and POS tag of the 3 neighbouring words before and after the current word token are also considered.
- **Dependency Relation Features.** We make use of the typed dependency relations returned by the Stanford Parser. The dependency relations provide a representation of grammatical relations between words in a sentence. We explore three types of features including the lemma (H-L) and PoS tag (H-PoS) of the head token in one dependency relation, and dependency labels (DL).
- **Co-occurrence (Co-oc) Feature.** Some cue keywords tend to co-occur in the sentences. For instance, core epistemic adjectives (e.g., *possible*) and verbs (e.g., *suggest*, *indicate*) often co-occur with modal auxiliaries, e.g., *may*, and *might*, to act as strong speculative cues as shown in (E2). Their co-occurrence might be a useful indicator to predict their speculative context.

The cue word classification model is constructed using the Conditional Random Fields (CRFs) [10] algorithm, implemented by the CRF++ Package[6]. The training data for the building of the classifier consists of a set word token instances, each of which contains a feature vector that is made up of four groups of features described above together with a cue class label – BIO tags. Due to space limitations, details about the CRF-based learning model can be found elsewhere[7].

C. Extracting uncertainty cues

Uncertainty cues were extracted from tag sequences labelled with the *BIO* scheme. If the cue contains multiple words, the tokens marked with the *B* or *I* tag are combined according to a tag priority, i.e. $B > I > O$. The extraction procedure is executed using the following rules:

- The first token of the cue starts with the token tagged as *B*. If it is not found, look for the *I* tag. If a word token is marked as *I*, and its previous token is tagged with *O*, then this token is considered as the beginning of a cue.
- The cue ends when it meets either a token tagged with *O* or a token started with a new tag *B* in which this token is regarded as the start point of a new cue.

D. Identifying speculative sentences

Once the system has extracted the uncertainty cues in sentences using the aforementioned CRFs model, we performed a separate post-processing step, using string matching to recognise infrequent multi-word cues that cannot be detected by the statistical machine learning model due to data sparseness. A small set of infrequent multi-word cues (e.g., *look like*, *to confirm*, *not clear*) were collected from the training data, and then used for the string matching-based cue recognition. If a sentence contains one or more extracted uncertainty cues, then it is classed as *speculative.*

[6] http://crfpp.sourceforge.net

[7] http://crfpp.googlecode.com/svn/trunk/doc/index.html#usage

IV. Uncertainty Scope Resolution

As described earlier, the scope of uncertainty may vary, and may be evident from certain linguistic properties. In our approach to scope resolution we rely heavily on syntactic information, taken from the phrase parse tree generated by the Stanford Parser. We also use various additional features related to specific syntactic constructions displayed by distinct uncertainty cue categories.

A. Heuristics to capture the scope of uncertainty

Following the BioScope guidelines for scope annotation [17], we manually inspected the gold standard of the BioScope corpus[8] and defined a set of heuristics which we developed to determine the scope of uncertainty. We list them below, organised by the categories of uncertainty cues given in Section II.

1. Auxiliaries

The scope of modal verbs differs in the *passive/active* voice of the main verb that follows the modal.

- If the main verb is *active*, then the scope starts with the modal and all its dependents, as in (E21).
- If the main verb is *passive*, then the scope includes the subject argument that appears in the preceding text, (E22).

E21. *A ROW Project Agent [may take on a responsibility of a discipline supervisor].*
E22. *[This function may be performed by a contractor or the ROW Engineer].*

2. Epistemic verbs

The scope for an epistemic verb depends on three main factors: (1) *passive/active* voice of the verb; (2) whether or not it is a *raising* verb (e.g., *seem*, *appear*); (3) if the subject argument of the verb is an *expletive* pronoun (e.g., *it*).

- If the verb is non-raising and active, scope starts at the cue word and spans to the end of the clause or sentence (E23).
- If the verb is passive, or is a raising verb, the scope includes the subject argument of the verb (E24) and (E25) respectively.

E23. *The multiple media requirement [implies the ability to generate a seamless map background from more than one CD ROM].*
E24. *Insulin is only delivered in circumstances where [it appears that the level is likely to go outside this range].*
E25. *This determination requires figuring out [what the CBS is supposed to do, i.e., the full set of features].*

3. Epistemic adjectives

- If the adjective has an *attributive* function, the scope covers the modified noun phrase and all its descendants (E26).
- If the adjective has a *predicative* function, the scope covers the subject argument of the head verb (the copula), as well as all its dependents (E27).

E26. *This function lists [possible issues that may prevent the undertaking of acquisition and steps to resolve those issues].*

[8] http://www.inf.u-szeged.hu/rgai/bioscope

E27. *[Job seekers will also be likely to use the custom home page feature].*

4. Epistemic adverbs

The scope for an adverb depends on the properties of the head word that it modifies.

- If the head word is a noun, the scope takes over the head word with all its (non-subject) syntactic descendents (E28).
- If the head word is a verb, the scope includes the subject argument of the head verb (E29).

E28. *The facilities, hardware, software, firmware, procedures, and documentation necessary to perform qualification, and [possibly other, testing of software].*

E29. *[Title Database internal to the system possibly linked to the Recorders Office].*

5. Epistemic nouns

The scope of epistemic nouns starts with the noun cue and includes its possible clause complement with *that*, (E30), or its indefinite complement with *to* (E31).

E30. *On the [assumption that the implementation will compute the GHA only once], Table 3 summarises the floating point operations required for each algorithm component.*

E31. *There is the [possibility to set the display text either in English or in German].*

6. Conjoining phrases

- If the conjoining phrase occurs as a single unit (e.g., *whether or not*) then the scope covers its argument and all its dependents (E32).
- If the cue is *or* and it co-occurs with another cue (e.g., *either*), the scope should combine them together (E33).

E32. *If the owner has made a counter offer, the ROW Chief decides [whether or not to accept the counter offer].*

E33. *Web sites hosting software are maintained [either by the LMI department or by another state agency].*

B. Automatically identifying scope

Our approach to scope resolution relies on the dependency structure information obtained from the phrase parse trees generated by the Stanford Parser, and the set of linguistic heuristics described in the previous section IV.A. These help identify the scope of each detected cue. We also impose the following constraints when annotating the uncertainty cues and their scope:

- Each cue should have one scope, and the cue must be contained in the corresponding scope.
- One scope probably have one or more cues, as in (E34).
- Cues and scopes have to be continuous.

E34. *The user [may specify whether to search for the term within the name of the information resource or the definition or both].*

We present a simple example to illustrate how the system uses the phrase parse tree and linguistic heuristics to determine the scope of the uncertainty cue '*may*' in the sample sentence (E21). Figure 2 shows the information that describes the properties of the uncertainty cue '*may*', and the search pattern on the parse tree which gives the required scope. Figure 3 shows the parse tree for the sample speculative sentence (E21) generated by the Stanford Parser, with each word token (leaf node) of the sentence tagged with its part of speech.

A. Uncertainty Cue
- Cue type: Auxiliary
- Cue term: *may*
- Cue headword: *may* (POS tag: *MD*)
- Cue property: *take* (main verb, active voice)

B. Searched Scope
- Search context: right context
- Search depth (layer): 2 (upper layer), and *VP* Node (start from the headword leaf node)

Figure 2. The frame-based representation of an uncertainty cue and searched scope for the example (E21).

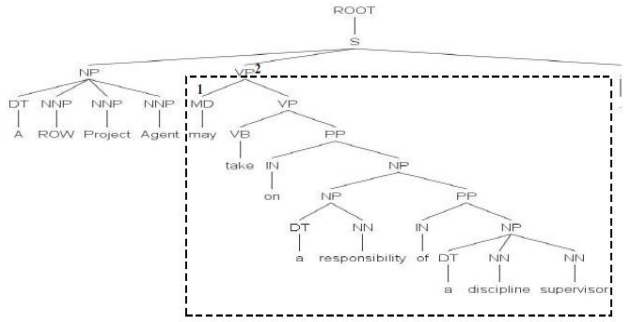

Figure 3. The phrase parse tree for the example (E21) by the Stanford Parser. The area in the rectangle with blue broken lines is the chunk subtree used as the scope of the uncertainty cue '*may*'.

Given an uncertainty cue, the system first checks a variety of syntactic properties associated with the cue, e.g., POS, argumenthood, voice, etc, as shown in Figure 2. It then decides the search strategy based on the context type (left or right context) and search depth in the parse tree. In the case of (E21), to find the scope of the uncertainty cue '*may*', the system begins the search from the start-point, i.e. the leaf node of the cue headword '*may*', and then tracks upward to the ancestor node *VP* (Node 2 in Figure 3) in the second layer. The chunk subtree (the rectangle area with broken lines in Figure 3) associated with the located ancestor node (Node 2) is trimmed from the whole tree as the scope of the uncertainty cue. As shown in Figure 3, the uncertainty scope is clearly defined, and is captured by the analysis of the syntactic role of the uncertainty cue given by the parse tree.

The search depth determines the scope of an uncertainty cue. In our system, the search depth for each cue is decided empirically, depending on several factors, such as the POS tag of the cue headword, the class type of the cue, and the linguistic properties of the cue (e.g., whether the scope needs to include the subject argument). In general, the search depth for various types of cues is limited to 2-4 upper layers starting from the leaf node of the headword.

As described above, the system usually starts a scope search from one of the leaf nodes; i.e. some word contained in the cue, in the parse tree. Hence, in the case of multi-word cues (e.g., *whether or not*, *may or may not*, and *not clear*,

etc.), the system firstly needs to determine the headword of the cue, e.g., *whether or not* → *whether*, *may or may not* → *may* (first one), and *not clear* → *clear*, before searching the phrase parse tree for the resolution of uncertainty scope.

C. Determing scope boundary

The previous section described how to ensure the scope contains the cue and the main elements of the scope. It is possible for the scope to extend beyond the points defined in part B. Generally, the default scope is set to start at the cue word and span to the end of the clause or sentence. However, as analysed in Section IV.A, in some special cases the subject argument can occur before the cue term in the text, and still be included in the scope. Moreover the dependents connected to the cue are sometimes attached to non-speculative information, such as when a secondary clause follows the primary clause containing the uncertainty cue. Consider the sentence (E35) below. Here, the scope for the cue '*would*' should begin with the noun phrase, '*a problem with this interface*', acting as the subject argument for the *passive* verb '*detected*', and the end scope will be terminated before the sentence-level conjunction '*until*'.

E35. *Normally, [a problem with this interface would not even be detected] until after the system was implemented.*

Due to the presence of the clause ending word '*until*' between the cue term '*would*' and the end point of the scope '*implemented*', the original end of the scope has to be modified. The end of the scope is moved forward to the word '*detected*' before the clause ending word '*until*'. In our system, we use a set of potential clause ending terms (e.g., *whereas*, *but*, *although*, *nevertheless,* etc.*)*. These words usually imply the beginning of a new clause in the sentence, and thus are considered as the scope ending signals.

Given an uncertainty cue, the system outputs a scope fragment extracted from the corresponding speculative sentence. However, as mentioned before, for some special cue pairs such as '*either…or*', and '*whether…or*', although they are two separate cues, they should form a single scope when they appear in the same sentence according to the annotation guideline. An additional heuristic was employed to ensure that the scopes for such conjunction cues are the same.

V. Experiments and Results

Our system performance was evaluated in terms of Precision (P), Recall (R), and their harmonic mean, the F-measure (F):

$$R = \frac{TP}{TP + FN} \qquad P = \frac{TP}{TP + FP} \qquad F = \frac{P \times R}{P + R}$$

where TP (true positive) is the number of correctly identified instances, FN (false negative) is the number of instances not identified by the system, and FP (false positive) is the number of instances that are incorrectly identified by the system.

As described earlier, uncertainty detection in requirements documents in practice consists of two tasks: (a) the identification of speculative sentences, each of which contains one or more uncertainty cues, and (b) in-sentence uncertainty scope detection, which aims to recognise speculative text spans inside the sentences.

For the speculative sentence identification task, the results reported here were obtained by performing 10-fold cross validation experiments on the requirements from our manually-annotated dataset. In each iteration, we trained on 90% and tested on 10% of the remaining data.

For the task of uncertainty scope determination, as described earlier, we used the gold standard of the BioScope corpus as training data to extract a set of linguistic heuristics for scope detection. We used the rules on our RE uncertainty dataset as the test data and report two sets of experimental results using different sources of uncertainty cues.

A. Speculative sentence identification

In uncertainty cue identification, evaluation is based on exact-match counts for uncertainty cues (possibly spanning multiple tokens). We also evaluate the percentage of speculative sentences that are correctly identified by the system.

(a) Impact of feature types on the CRF-based cue classifier

To evaluate the contribution of the various features for uncertainty cue identification, we performed a series of experiments in which different sets of linguistic features are added to the word-token feature baseline and new classifiers are trained. We used word-token features as the evaluation baseline because of the important role that these features play in identifying uncertainty cues. We compared the performance of these new classifiers to the word-token feature baseline.

Table III shows the impact of different feature sets on the performance of uncertainty cue identification. The context-related features exhibit strong distinguishing capability, which substantially improve the performance with an increase of 7.1% in precision and 9.6% in recall. However, with the addition of the dependency relation features and co-occurrence features, the overall F-measure average improves only slightly by 1%. We notice that when all the features work together in combination, the system would achieve the highest performance for cue classification. This suggests that a wide variety of features indeed characterise different aspects of uncertainty cues.

TABLE III. The impact of feature types on the performance of uncertainty cue identification (DR-Dependency Relation)

	P (%)	R (%)	F (%)
Word	77.18	67.38	72.00
Word + Context	84.29	76.98	80.47
Word + Context + DR	85.43	76.70	80.83
All features	85.58	77.65	81.42

(b) Performance of different cue types

Table IV summarises the performance of our system in predicting different types of uncertainty cues. In comparison to other types of uncertainty cues, adverb cues perform best and achieve an F-measure as high as 90.43%. One possible explanation for this is because adverbs are much more straightforward to recognise than other types of uncertainty cues. Auxiliaries, the most common cues in our uncertainty dataset, also perform well and the F-measure score reaches

84.68%. The performance on the identification of conjunction cues is worse, and the recall drops to 49.6% only. This is caused primarily by the poor recognition of '*or*', which is very ambiguous when it functions as the uncertainty cue. An example of '*or*' as an uncertainty cue is shown in (E36), and as a non-uncertainty cue in (E37). There are 1320 occurrences of '*or*' in our dataset, and only 64 of them are uncertainty cues. The system predicts 58 occurrences of '*or*' as uncertainty cues, and only 23 are correctly recognised.

E36. *Quickly pressing a mouse button in order to make a selection or give a command.* [Uncertainty cue]

E37. *The radio programs can be changed with the steering wheel buttons left or right.* [Non-uncertainty cue]

TABLE IV. THE PERFORMANCE ON THE IDENTIFICATION OF DIFFERENT TYPES OF UNCERTAINTY CUES

	P (%)	R (%)	F(%)
Auxiliaries	89.33	80.49	84.68
Verbs	88.82	84.48	86.60
Adjectives	73.11	79.09	75.98
Adverbs	96.30	85.25	90.43
Nouns	64.86	68.57	66.67
Conjunctions	78.75	49.60	60.86

TABLE V. THE OVERALL PERFORMANCE ON THE CUE DETECTION LEVEL AND THE SPECULATIVE SENTENCE IDENTIFICATION LEVEL

	P (%)	R (%)	F (%)
cue level	85.58	77.65	81.42
sentence level	88.19	83.16	85.60

Table V shows the overall performance of uncertainty identification at the cue level and the sentence level. It is not surprising that the performance on speculative sentence identification is better than that of cue recognition. Part of the reason for this is because speculative sentence identification is much more tolerant of partially recognised cues in the case of multi-word cues. In cases where only part of a multi-word cue is recognised but the whole cue is missed, the sentence can still be correctly identified as speculative.

B. Uncertainty scope detection

In the scope resolution task, a scope is considered to be correct if it has the correct start and end points in the sentence and is associated with the correct cues. Given an uncertainty cue, the system searches the corresponding sentence and locates the speculative text spans invoked by the cue. Here we report on the performance using two sets of cue input, one with gold standard cues (i.e. the manually annotated cues) and the other with the cues predicted by our cue recognition model, in order to compare the effect of cue identification at the earlier stage on the performance of scope detection at the later stage.

Table VI shows that performance of scope identification using gold standard cues and predicted cues, respectively. With gold standard cues as input data, the system performs well on the whole dataset, with an overall F-measure of 61.63%. This suggests that the Stanford Parser's phrase parse tree essentially captures the necessary syntactic structure of the sentence. With the help of the hand-crafted syntactic heuristics described in Section IV.A, the parse trees can be used to predict correctly the speculative text span invoked by the uncertainty cue. The system works well for four particular types of uncertainty cues: Auxiliaries, Verbs, Nouns, and Conjunctions, and all of these four cue types have an F-measure score of approximately 62%. However, in the case of Adverb cues, the system performance significantly deteriorates and the F-measure score decreases to only 40.98%. One of the main reasons for this is weak capability to detect clause structures in more complicated sentences. These errors are caused by incorrect parse trees being generated by the Stanford Parser.

TABLE VI. THE OVERALL PERFORMANCE ON UNCERTAINTY SCOPE IDENTIFICATION

	Gold Standard Cues			Predicted Cues		
	P (%)	R (%)	F (%)	P (%)	R (%)	F(%)
Auxiliaries	65.23	66.30	65.76	60.78	55.62	58.08
Verbs	62.36	62.36	62.36	56.63	54.02	55.29
Adjectives	52.73	52.73	52.73	39.66	41.82	40.71
Adverbs	40.98	40.98	40.98	37.04	32.79	34.78
Nouns	61.11	62.86	61.97	37.84	40.00	38.89
Conjunctions	61.78	61.78	61.78	62.19	40.47	49.03
Overall	61.25	62.01	61.63	54.37	49.95	52.07

To investigate how well the system performs under the more natural but less-than-perfect condition of automatic recognition of uncertainty cues, we conducted another set of experiments using the cues produced at the earlier cue identification stage. Unsurprisingly, the overall system performance drops 9.6 percentage points in F-measure due to imperfect behavior in cue identification. The system performs poorly especially for the cues, *adjectives*, *adverbs*, and *nouns*, with an F-measure of less than 40%. This indicates that there is still much room for improvement in scope identification.

VI. THREATS TO VALIDITY

This section discusses potential threats that might affect the validity of our work and the uncertainty detection by our system, and how they are mitigated or accommodated.

A. Errors analysis

Our error analysis on false positives and negatives reveals that one of the most important sources of error is caused by incorrect analyses by the Genia Tagger and Stanford Parser. Despite common recommendations to the contrary, the requirements in our dataset included many long, complex sentences, and often left the tagger and parser unable to correctly analyse the text, leading to misinterpreted lexical or syntactical ambiguities. Another potential negative factor is the incomplete nature of our source corpus, that is, the five cue categories are not comprehensive and complete, especially for the instances that are previously unseen in our annotated dataset. This could lead to some possible false negatives in cue identification in terms of new requirements datasets. Moreover, multi-word cues were found by the naive string matching approach that probably results in an overfitting problem when the system processes some new requirements dataset.

B. Uncertainty strength

While recognising whether a sentence is speculative or not is useful, it seems more interesting and clearly much more challenging to determine the strength of the uncertainty. It is clear that not all speculative requirements are equally strong and that the choice of speculation device affects the strength of the uncertainty. From the point of view of evaluating quality of requirements expressiveness and of prioritising the effort spent on inspection and further elicitation, it would be useful to quantify the degree of uncertainty as an indication of the confidence that the author has in his or her proposition. However, determining the strength of a speculative requirement is not trivial. The main reason is due to the peripheral nature in uncertainty language, which results in low inter-annotator agreement in determining the strength degree of uncertainty [11]. Nevertheless, the study of how to represent uncertainty strength in speculative requirements will be an interesting topic in future work.

C. Applicability to requirements engineering practice

While we have demonstrated the effectiveness of our approach at identifying speculative sentences, our validation does not extend to the use of such information in requirements engineering practice. There is thus a threat that the identification of speculative sentences is in practice of no use, or too hard to use for practitioners.

Discounting this threat will require further studies in the field. However, we are confident that a suitable, low-cost (both in computational and in cognitive terms) integration with tools in common use (such as IBM Rational DOORS) is feasible. In this scenario, speculative sentences could be flagged in a non-intrusive way (e.g., by underlining with yellow squiggles the relevant parts), thus providing the analyst a subtle but unmistakable cue that something anomalous could lie behind the uncertainty. It would then be left to the analyst to decide what the follow-up should be (e.g., further questioning of stakeholders, dismissing as irrelevant, or explicitly stating that the final choice is left to the designers).

VII. Related Work

A. Quality control and analysis in NL requirements

In industrial practice, the vast majority of requirements documents are still written in natural language due to various reasons such as ease of expression and the convenience of communication between different stakeholders [13]. However, requirements quality also suffers from the typical NL problems such as ambiguity and uncertainty. A number of studies have been done in the past to address research issues concerning quality control and analysis in NL requirements. Here we briefly discuss some of them that we consider as particularly related to our research.

Several studies proposed a quality model (QM) to measure quality attributes in NL requirements documents by using a set of quality metrics (e.g., vagueness, subjectivity, optionality, weakness, etc.). They developed 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) [3] 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. Wilson et al. [18] 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. [4] 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 [7] made use of the semantic relationships between concepts on a domain ontology to check the ambiguity property of a requirements item, i.e. whether the requirements item is mapped into several concepts. Ben Achour et al. [1] discussed the vague and unverifiable problem in requirements, which is caused by ambiguity of words and phrases.

In our previous work, we developed an automated tool [19] to identify potentially nocuous ambiguity in NL requirements, which also adopted machine-learning to combine a set of heuristics with human judgments to build a nocuity classifier. In this paper we investigate another linguistic phenomenon, uncertainty, in NL requirements, in which the stakeholders use speculative language to express tentativeness and possibility when they verbalise their own needs. We described a CRF-based learning model for the recognition of uncertainty cues at the term level and a heuristics-based method for the determination of uncertainty scope with the help of sentence parse tree.

B. Uncertainty research

In recent years, uncertainty detection research has received considerable attentions in the natural language processing (NLP) community. Light et al. [11] first explored the possibility of automatically classifying sentences into speculative or non-speculative by looking for a list of collected specific keywords. Medlock and Briscoe [12] proposed a weakly supervised learning approach, which used single words as input features for hedging classification. A probabilistic model was exploited to acquire training data for the learning of hedge classifier. Szarvas [16] extended their work by introducing n-gram word features and developing a weakly supervised method for feature selection. Kilicoglu and Bergler [9] proposed a linguistic motivated approach that combined the knowledge from existing lexical resources and hand-crafted syntactic patterns. Additionally, they applied a weighted scheme to estimate the speculative level of the sentences. Ganter and Strube [5] proposed an automated approach to investigate the hedging problem in Wikipedia using tagged weasel words combined with syntactic patterns.

In this paper, we have, for the first time, investigated the uncertainty detection problem in requirements documents. We analysed several uncertainty types that frequently occur in NL requirements. We built on recent work [14, 15], exploring the use of supervised ML approaches to the detection of uncertainty cues, and we demonstrated that the success of a rule-based approach for the scope determination.

VIII. Conclusion and Future Work

The use of speculative language in NL requirements can indicate the presence of uncertainty – where misunderstandings by system developers might occur because the requirements have not been fully understood nor clearly expressed. By detecting uncertainty in NL requirements, the quality of those requirements can be evaluated in order to minimise adverse effects on the final software product.

In this paper, we showed that it is possible to automatically identify uncertainty in NL requirements by using a linguistically motivated approach. We manually annotated a requirements corpus with uncertainty cues. The annotated corpus was then used for developing and evaluating our automatic uncertainty identification system. We employed a supervised machine learning algorithm (CRFs) to recognise uncertainty cues from sentences using a wide variety of linguistic features. Our system performed well at the speculative sentence identification task, but was less effective at uncertainty scope identification (although we suggest that the latter task is less significant in practice, since a sentence will be presented to and reviewed by an analyst anyway).

There are a number of areas for future work. For example, we need to extend our work to estimate the degree of uncertainty in speculative requirements, to better understand the effect of uncertainty on requirements (mis)understanding. We also need to examine prediction accuracy of uncertainty cues: will different machine learning techniques improve the system performance?

The overall goal of our research has been to develop a series of automated tools that support the detection of linguistic problems in NL requirements, such as nocuous ambiguity detection tools [19], and the uncertainty identification tool described in this paper. Our aim is to help requirements analysts to inspect and analyse potentially problematic requirements to improve their quality.

Acknowledgment

We acknowledge the support of the UK Engineering and Physical Sciences Research Council (EPSRC) as part of the MaTREx project (EP/F068859/1), the Science Foundation Ireland (SFI grant 10/CE/I1855), and The European Research Council (ERC).

References

[1] C. Ben Achour, C. Rolland, C. Souveyet, and N. A. M. Maiden, "Guiding Use Case Authoring: Results of an Empirical Study," in *Proc. of the 7th IEEE Intl. Requirements Engineering Conference (RE'99)* 1999, pp. 36-43.

[2] A. de Roeck, R. Ball, K. Brown, C. Fox, M. Groefsema, N. Obeid, and R. Turner, "Helpful answers to modal and hypothetical questions," in *Proc. of the 5th conference on European chapter of the Association for Computational Linguistics (EACL'91)*, 1991.

[3] F. Fabbrini, M. Fusani, S. Gnesi, and G. Lami, "The linguistic approach to the natural language requirements, quality: benefits of the use of an automatic tool," in *Proc. of the 26th annual IEEE computer society—NASA GSFC software engineering workshop*, 2001, pp. 97–105.

[4] A. Fantechi, S. Gnesi, G. Lami, and A. Maccari, "Applications of Linguistic Techniques for Use Case Analysis," *Requirements Engineering,* vol. 8, pp. 161-170 2003.

[5] V. Ganter and M. Strube, "Finding hedges by chasing weasels: Hedge detection using Wikipedia tags and shallow linguistic features," in *Proc. of the ACL-IJCNLP 2009 Conference*, 2009, pp. 173-176.

[6] K. Hyland, *Hedging in scientific research articles*. Amsterdam: John Benjamins B.V, 1998.

[7] H. Kaiya and M. Saeki, "Using Domain Ontology as Domain Knowledge for Requirements Elicitation," in *Proc. of the 14th IEEE Intl. Requirements Engineering Conference (RE'06)* 2006, pp. 186-195.

[8] M. Jackson, Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices: Addison Wesley, 1995.

[9] H. Kilicoglu and S. Bergler, "Recognizing speculative language in biomedical research articles: a linguistically motivated perspective," *BMC Bioinformatics,* vol. 9, p. s10, 2008.

[10] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labelling sequence data," in *Proc. of the Intl. Conference on Machine Learning (ICML-2001)*, 2001, pp. 282-289.

[11] M. Light, X. Y. Qiu, and Srinivasan, "The language of bioscience: facts, speculations, and statements in between," in *BioLINK 2004: Linking Biological Literature, Ontologies and Databases*, 2004, pp. 17–24.

[12] B. Medlock and T. Briscoe, "Weakly supervised learning for hedge classification in scientific literature," in *Proc. of the 45th Meeting of the Association for Computational Linguistics (ACL'07)*, 2007, pp. 992–999.

[13] L. Mich, M. Franch, and P. Inverardi, "Market research for requirements analysis using linguistic tools," *Requirements Engineering,* vol. 9, pp. 40-56, 2004.

[14] L. Øvrelid, E. Velldal, and S. Oepen, "Syntactic Scope Resolution in Uncertainty Analysis," in *Proc. of the 23rd Intl. Conference on Computational Linguistics (Coling 2010)*, 2010 pp. 1379–1387.

[15] V. Prabhakaran, "Uncertainty Learning Using SVMs and CRFs," in Proc. of the 14th Conference on Computational Natural Language Learning: Shared Task, 2010, pp. 132–137.

[16] G. Szarvas, "Hedge classification in biomedical texts with a weakly supervised selection of keywords," in *Proc. of the 46th Meeting of the Association for Computational Linguistics (ACL'08)*, 2008, pp. 281–289.

[17] V. Vincze, G. Szarvas, R. Farkas, G. Mora, and J. Csirik, "The BioScope corpus: biomedical texts annotated for uncertainty, negation and their scopes," *BMC Bioinformatics,* vol. 9, p. S9, 2008.

[18] W. M. Wilson, L. H. Rosenberg, and L. E. Hyatt, "Automated analysis of requirement specifications," in *Proc. of the 19th Intl. Conference on Software Engineering (ICSE)*, 1997, pp. 161–171.

[19] H. Yang, A. de Roeck, V. Gervasi, A. Willis, and B. Nuseibeh, "Analysing Anaphoric Ambiguity in Natural Language Requirements," *Requirements Engineering,* vol. 16, pp. 163-189, 2011.