Noname manuscript No.
(will be inserted by the editor)

An NLP Approach for Cross-domain Ambiguity Detection in Requirements Engineering

Alessio Ferrari · Andrea Esuli

Received: date / Accepted: date

Abstract During requirements elicitation, different stakeholders with diverse backgrounds and skills need to effectively communicate to reach a shared understanding of the problem at hand. Linguistic ambiguity due to terminological discrepancies may occur between stakeholders that belong to different technical domains. If not properly addressed, ambiguity can create frustration and distrust during requirements elicitation meetings, and lead to problems at later stages of development. This paper presents a natural language processing approach to identify ambiguous terms between different domains, and rank them by ambiguity score. The approach is based on building domain-specific language models, one for each stakeholders' domain. *Word embeddings* from each language model are compared in order to measure the differences of use of a term, thus estimating its potential ambiguity across the domains of interest. We evaluate the approach on seven potential elicitation scenarios involving five domains. In the evaluation, we compare the ambiguity rankings automatically produced with the ones manually obtained by the authors as well as by multiple annotators recruited through Amazon Mechanical Turk. The rankings produced by the approach lead to a maximum Kendall's Tau of 88%. However, for several scenarios, the application of the approach was unsuccessful in terms of performance. Analysis of the agreement among annotators and of the observed inaccuracies offer hints for further research on the relationship between domain knowledge and natural language ambiguity.

A. Ferrari
CNR-ISTI, Via G. Moruzzi 1, 56124, Pisa, Italy
Tel.: +39 050 621 2806
E-mail: alessio.ferrari@isti.cnr.it

A. Esuli
CNR-ISTI, Via G. Moruzzi 1, 56124, Pisa, Italy
Tel.: +39 050 621 2890
E-mail: andrea.esuli@isti.cnr.it

Keywords Natural Language Processing · Requirements Engineering · Ambiguity · Domain Knowledge

1 Introduction

Software systems are developed for a large variety of domains, which range from non-technical areas, such as fitness or entertainment, to more technical ones, such as aerospace or electronic engineering. When developing a novel system for a specific domain – or even involving different domains at the same time (e.g., medicine and fitness) – requirements analysts need to interact with domain experts to elicit the domain knowledge needed to develop the system (Cleland-Huang 2015; Hoffman et al. 1995; Lian et al. 2016). To this end, requirements elicitation meetings in the form of interviews, focus groups, or workshops are organised with the stakeholders (Dieste and Juristo 2011; Zowghi and Coulin 2005). During these meetings, requirements analysts and domain experts may use their own specialised jargons, and ambiguity may occur between analysts and domain experts, as well as between stakeholders belonging to different domains. As well known, unresolved ambiguities in the early phases of the software process may cause costly problems in later stages of the development (Fernández et al. 2017).

Several works have been proposed in the literature to tackle the problem of ambiguity, with a main focus on written requirements. Part of the works focuses on the identification of terms and expressions that may cause ambiguity, such as the term "all" and plural words (Berry and Kamsties 2005; Tjong and Berry 2013), adjectives and adverbs (Femmer et al. 2017; Gleich et al. 2010; Rosadini et al. 2017; Tjong and Berry 2013), and passive voice (Femmer et al. 2014, 2017). Other works focus on syntactic ambiguities, which are caused by ambiguous sentence structures (Berry and Kamsties 2004). In particular, tools have been provided for the detection of coordination ambiguities, i.e., ambiguities triggered by "and" or "or" conjunctions (Chantree et al. 2006) and anaphoric ones, i.e., ambiguities triggered by pronouns (Yang et al. 2011). All these works deal with domain-independent ambiguous terms and constructions. Domain-dependent ambiguity received less attention in previous work, although domain knowledge has been identified among the most frequent sources of ambiguity, especially in requirements elicitation interviews (Ferrari et al. 2016).

In this paper, we present an approach to support the identification of potentially ambiguous terms in the context of requirements elicitation interviews and group meetings involving different domains. Given a set of domains, the approach produces a ranked list of terms, scored by their degree of cross-domain ambiguity. The proposed approach is based on *word embeddings* (Mikolov et al. 2013), which are semantic-laden word representations that are automatically learned based on a natural language (NL) corpus given as input. Based on different corpora automatically crawled from Wikipedia portals, we build different domain-specific language models, i.e., vector spaces of word embeddings.

We first select terms that frequently occur in groups of domains – e.g., terms that are frequently used in computer science and may occur also in medicine, such as *formula* or *mouse*. We call these terms *dominant shared terms*. Then, we estimate the variation of meaning of these terms in the different domains by comparing the lists of the most similar words in each domain-specific language model. The ambiguity ranking is finally produced based on the ambiguity potential of the terms.

To evaluate the proposed method, we test it on a set of seven hypothetical, yet realistic scenarios concerning the development of different products, and involving combinations of stakeholders from five different domains, namely Computer Science, Electronic Engineering, Mechanical Engineering, Medicine and Sports. For each scenario, we produce a list of terms ranked by cross-domain ambiguity. The evaluation of the rankings is performed in two iterations. In a fist iteration, the rankings are compared with those produced on the basis of a manual annotation performed by the authors. In a second one, they are compared with the rankings produced through a Human Intelligence Task (HIT) submitted through Amazon Mechanical Turk (MTurk). A maximum Kendall's Tau of 88% is obtained, confirming the potential of the approach for part of the scenarios. Lower values were observed for a subset of the scenarios, as well as a low agreement among annotators. We perform a qualitative evaluation of the results to account for inaccuracies and notable cases of disagreement. Our analysis shows that several factors had an impact on the results, including: the existence of high-level terms, as, e.g., *consequence*, and named entities, e.g., proper nouns, which do not vary their meaning regardless of their linguistic context in the domain-specific corpora; the presence of terms that are used with different meanings also within domains; problems related to the process adopted for evaluating the method, as, e.g., the interpretation of instructions by the human assessor as well as their lack of domain knowledge. The source code of our approach, together with the data used for evaluation, as well as the support algorithms to process the data are available in our repository[1].

This paper extends a contribution to the AIRE'18 workshop (Ferrari et al. 2018a). With respect the original work, the current one adds the following relevant content: (a) an improved approach to compute the set of dominant shared terms (Sect. 3.3); (b) a systematic evaluation of the proposed method (Sect. 4); (c) a thorough review of related work (Sect. 5).

The remainder of the paper is structured as follows. In Sect. 2 we provide some background on ambiguity and word embeddings, and we summarise our previous work. In Sect. 3 we outline the proposed approach. In Sect. 4 we present the research design adopted for the evaluation of the method and we discuss the results. Sect. 6 provides final remarks.

[1] `https://doi.org/10.5281/zenodo.1476902`

2 Background

2.1 Ambiguity Categories in Requirements Engineering

Ambiguity is largely studied in written NL requirements, and, although our main focus is on requirements elicitation meetings performed in spoken NL, it is useful to refer to established classifications of ambiguity in written requirements. Ambiguities in NL requirements are normally classified into four main categories (Berry et al. 2003): *lexical*, i.e., the terms used have unrelated vocabulary meanings; *syntactic*, i.e., the sentence has more than one syntax tree, each one with a different meaning; *semantic*, i.e., a sentence can be translated into more than one logic expression; and *pragmatic*, i.e., the meaning of the sentence depends on the *context* in which it is used. The term *context* is used as a general concept, which includes different levels (Ferrari and Gnesi 2012): (1) those sentences immediately preceding and following the current one, (2) the other sentences placed in other sections of the document, (3) the *domain knowledge* of the subject reading the requirement (the reader), (4) the reader's *common sense knowledge*, and (5) the reader's *viewpoint*.

Berry et al. (2003) consider also *language errors* as a separate class of ambiguity. In addition, two other phenomena closely related to ambiguity are discussed in the literature (Lami et al. 2001), namely: *vagueness*, i.e., an expression is vague if it admits borderline cases, which are cases in which the truth value of the expression cannot be decided – normally due to adjectives or adverbs; *generality*, i.e., an expression is general if it needs to be specified with more details to make sense of it. Examples for each category are reported in Berry et al. (2003) and Ferrari et al. (2016).

As highlighted, among others, by Massey et al. (2014), real-world ambiguity cases may fall in more than one category. The cases of *domain-dependent ambiguity* that we wish to study in this paper fall in the categories of: (1) lexical ambiguity, since some words may be used with different vocabulary meanings in different domains – e.g., the term *windows*: operating system or glass openings of a vehicle?; (2) pragmatic ambiguity, since the interpretation of a word may depend on the domain-specific background of the reader – e.g., *machine*: a software system or a specific medical system for diagnostic support?; and (3) generality, since, in some cases, the actual domain-specific meaning of a word may become clear when the word is specified better – e.g., *interface*: software or hardware interface?

2.2 Word Embeddings

The term *word embeddings* encompasses a series of techniques for representing the meaning of a word in a dense numerical vector in a vector space. Given these representations, the semantic similarity among words can be computed by measuring the similarity between vectors. Word embeddings are based on the distributional hypothesis of Harris (Harris 1954), which states that words

that appear in similar linguistic contexts have similar meanings. Or, using the terms of the linguist J. R. Firth, the meaning of a word is given *by the company it keeps* (Firth 1968).

Among the various word embeddings techniques, one of the most widely known was proposed by Mikolov et al. (2013). This technique is known as skip-gram with negative sampling (SGNS), and is implemented in the software package `word2vec`. The name `word2vec` is also commonly used to refer to the technique. The actual approach adopted by `word2vec` has been subject of several studies in the NLP field, and one theoretical explanation is provided by Levy and Goldberg (2014). We refer to this work for more details. Here, we outline the fundamental concepts of the technique, to give the spirit of the approach.

The SGNS variant of `word2vec` learns word embeddings as a by-product of training a two-layers neural network on the task of predicting from a single *input word* w a set of *context words* c_w. Given a corpus of documents, large amounts of training examples can be easily generated. All the words appearing in the corpus of documents define a vocabulary V. Any occurrence of any word in the corpus can be used as an input word w, while the context of that occurrence is defined by the words surrounding it in the corpus, considering a window of size L, i.e., $c_w = \{w_{i-L}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+L}\}$. For example, if the corpus contains the fragment *interviews are used to elicit requirements*, the context of size 2 for the word $w =$ *used* is $c_w = \{$*interviews*, *are*, *to*, *elicit*$\}$.

Each $w \in V$ is associated with a vector $\mathbf{w}$, where $\mathbf{w} \in \mathbb{R}^d$, and d defines the embedding space dimensionality. These vectors are the ones that the `word2vec` algorithm aims to produce from the corpus.

In detail, the skip-gram network is fed in input with a word w encoded as a one-hot vector $\mathbf{h}_w$ on the vocabulary V. The set of words that appear in text next to w, i.e., the context c_w, is similarly encoded as a vector $\mathbf{c}_w$ on the vocabulary V, with ones only in the positions associated to words in c_w. The vector $\mathbf{h}_w$ acts as a row selector on a word embeddings matrix W of size $|V| \cdot d$ where d defines the length of the embeddings. The selected embedding $\mathbf{w} = W^T v_w$ is then multiplied to a second matrix W' of size $d \cdot |V|$, producing $\mathbf{u} = W'^T \mathbf{w}$. The softmax function is applied to $\mathbf{u}$ producing the final prediction of the probabilities of words to appear in the context of the input word $\tilde{c}_y = \text{softmax}(\mathbf{u})$, which is then compared to $\mathbf{c}_w$ so that the matrices W' and W can be updated through backpropagation. This process is repeated by extracting all the possible word-context pairs from the corpus.

The generated language model defines an embedding vector space for the words in the processed corpus, and have the relevant property that words that have similar meanings, or are related, in the input corpus are represented through vectors that are similar, i.e., closely placed in the vector space. Depending on the input corpus, different language models are generated that characterise the similarity between words in the specific corpus. Hence, by building corpora from different domain-specific documents, we can compare the meaning of a word in the different domains.

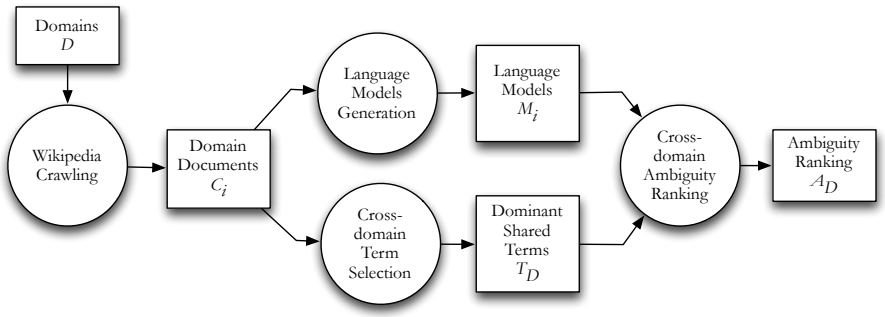

Fig. 1: Approach to Measure Cross-domain Ambiguity

2.3 Detecting Domain-dependent Ambiguity

In a previous work, we proposed the first approach to address domain-dependent ambiguities, and we defined a technique based on word embeddings to estimate the ambiguity potential of typical computer science (CS) words when they are used in different domains (Ferrari et al. 2017b). Specifically, we proposed to compare the similarity between a frequent CS word, such as *system*, *data*, *user*, *etc.*, in a language model constructed with CS-specific documents, and the same word when the language model was enriched with domain-specific documents. In this way, we studied how the meaning of common CS words varies when these words are used in different domains. However, the approach did not consider the ambiguity that could be triggered by domain-specific words outside the CS domains. In other terms, the approach did not account for those situations in which ambiguities were raised by terms used by domain experts. Furthermore, the approach had some technical limitations, due to (a) the need to construct a language model for each combination of domains, and (b) the need to modify the domain-specific documents.

Our initial idea was later developed with a general approach that, given a set of domains, produces a ranked list of potentially ambiguous terms, based on a list of *dominant shared terms* (Ferrari et al. 2018a). These are the words that are *common* between the different domains of the set, and that are *frequent* in the domains. Intuitively, these words are those that, if they have diverse domain-dependent meanings, are more likely to be the source of ambiguity during a conversation between stakeholders with distant domain backgrounds. The proposed approach also addressed the technical limitations of the previous work. The current paper extends Ferrari et al. (2018a) by tackling the two main points of the future work as planned in the original paper, namely:

1. Refinement of the procedure for the selection of dominant shared terms: our initial approach considered as dominant shared terms solely those that occurred in *all* the domains, while ambiguity may occur between pairs or subsets of domains involved in a meeting. The current procedure takes into account this aspect (see Sect. 3.3);

2. Systematic evaluation of the approach: the initial proposal was showcased on relevant examples, while here we perform a structured evaluation with quantitative and qualitative analyses (Sect. 4).

3 Approach

In Fig. 1 the proposed approach to produce a ranked list of potentially ambiguous terms is depicted. In our approach we focus on *nouns*, instead of terms in general, to restrict our scope. However, the approach can be similarly applied to other parts of speech.

The approach is as follows. We first crawl Wikipedia to extract domain specific documents for a given domain (**Wikipedia Crawling**). Then, we apply the `word2vec` algorithm on the corpus composed by the domain-specific documents to learn the word embeddings (**Language Models Generation**). Then, we search for the most frequent *nouns* that occur between sets of corpora (**Cross-domain Term Selection**). These nouns, which we name *dominant shared terms*, are those whose meanings we wish to compare across the different domains. As mentioned, these are the words that are commonly used in more than one domain, and thus they may cause frequent misunderstandings when they are used with different meanings by the stakeholders. Finally we measure the degree of ambiguity of the dominant shared terms, and we provide a ranking based on this measure (**Cross-domain Ambiguity Ranking**). This is achieved by exploiting the vector space of `word2vec`'s embeddings as a similarity space, in which the closer two embeddings are (as measured by cosine similarity) the more similar/related the two relative words can be considered. For each word, `word2vec` can return a ranked list of its most similar words in the embeddings' space of the language model, together with their similarity value. Hence, given a dominant shared term, we compare the lists of its most similar words determined by different domain-specific language models, and, in this way, we aim to estimate its degree of ambiguity. The more words the two lists have in common, and the closer are the similarity values for the same words in the two lists, the less ambiguous the dominant shared term that produced the lists can be considered.

The different steps are described in the following subsections. The first step is analogous to our previous work, and we provide only a brief summary of it.

3.1 Wikipedia Crawling

Given a set $\mathcal{D} = \{D_i : i = 1 \dots n\}$ of n domains, the Wikipedia Crawling step produces one corpus C_i for each domain D_i in the set. Each C_i includes pages from a domain-specific portal of Wikipedia. Each Wikipedia portal is structured as a tree in which nodes are categories, and leaves are pages.

To have a visual example, the reader is suggested to access the root category of the portal for CS[2]. To retrieve the pages indexed by a Wikipedia portal, we implemented an algorithm that, given the root category of the portal, performs a breadth-first search on the subcategories of the root category, and retrieves all the Wikipedia pages that are reachable through the search. Since the number of reachable pages may be extremely extensive, we set a maximum threshold of $10,000$ pages to be downloaded from each portal. In addition, we set a maximum depth of 2 for the subcategories to be explored. Focusing on higher-level subcategories allows reaching pages that are more general in terms of content with respect to deeper subcategories.

We then pre-process each corpus – ancillary step not reported in Figure 1 – by (a) transforming each word to lowercase (b) removing stop-words and by (c) lemmatizing the words, i.e., by reducing each word to its lemma. We remove stop-words, i.e., common words such as conjunctions, articles and pronouns, since in this context we are not interested in these words, which do not bear a domain-specific meaning. Lemmatization allows the treatment of word inflections (e.g., *requirements* and *requirement*) as a single word (i.e., *requirement*). After pre-processing, a sentence such as *interviews are used to elicit requirements* is transformed into *interview use elicit requirement.*

3.2 Language Models Generation

In this step, domain-specific language models M_i are learned by means of the `word2vec` algorithm (Mikolov et al. 2013), based on each input corpus C_i. The algorithm requires to define the value of d, i.e., the embedding's dimensionality; the value of L, i.e., the length of the context to observe; the value m of the minimum number of occurrences that a word should have to be considered by the algorithm. In our case, we set $d = 50$, $L = 10$, $m = 10$. These values have been selected based on preliminary experiments on the data.

3.3 Cross-domain Term Selection

This step produces a set $T_{\mathcal{D}}$ of dominant shared terms between the domains in $\mathcal{D}$. The pseudo-code[3] of the algorithm used to produce the set is reported as Algorithm 1.

The algorithm takes as input the set of domain specific corpora $\mathcal{C} = \{C_i : i = 1 \dots n\}$, as well as two numerical parameters, k and ρ. The parameter k is the minimum number of occurrences that a term should have in at least one domain to be considered as a candidate to be included in $T_{\mathcal{D}}$. Indeed, if a certain term is not sufficiently frequent in at least one domain, it cannot be considered as part of the dominant shared terms. The parameter ρ is a frequency ratio, and indicates how frequent the term should be in at least one

[2] `https://en.wikipedia.org/wiki/Category:Computer_science`

[3] In the pseudo-code, array indexes start from 1.

```
SELECT-TERMS(C, k, ρ)
1  T_D ← ∅
2
3  for C_i ∈ C
4        do for t ∈ VOCABULARY(C_i)
5              do if FREQ(t, C_i) ≥ k and t ∉ T_D
6                    do for C_j ∈ C \ C_i
7                          do if FREQ(t, C_j) ≥ ρ × FREQ(t, C_i)
8                                do T_D ← T_D ∪ {t}
9  return T_D
```

Algorithm 1: Selection of terms to be tested for ambiguity.

other domain to be included in $T_{\mathcal{D}}$. If a certain term is highly frequent in a certain domain, but no other domain exists in which the term is sufficiently represented, then the term is not considered as part of $T_{\mathcal{D}}$. For each corpus, the algorithm scans all the terms included in the vocabulary of the corpus (procedure VOCABULARY), and considers solely those whose frequency (FREQ) is greater or equal than k. Then, the algorithm checks whether another domain exists for which the term is frequent at least ρ times the frequency in the original domain. If the term passes these checks, it is added to the set $T_{\mathcal{D}}$.

With this approach, we produce a list of terms that (1) are common to at least two domains of the set, (2) are highly frequent in at least one domain, and (3) appear sufficiently often in at least one other domain.

3.4 Cross-domain Ambiguity Ranking

The step produces a ranked list $A_{\mathcal{D}}$ of the terms in $T_{\mathcal{D}}$, in which the ranking depends on the degree of cross-domain ambiguity of the terms, estimated according to the language models $\mathcal{M} = \{M_i : i = 1 \dots n\}$.

The idea of the algorithm to produce this ranking is as follows. For each dominant shared term, we compare the lists of its most similar words in the different domains, generated according to the `word2vec` language models of each domain. By comparing domain-specific lists of most similar words (in the following, *similarity lists*), we are able to estimate the variation of meaning of the dominant shared term. The comparison among the lists is performed by computing a dissimilarity score that we deem to be strictly correlated to the ambiguity of the word across the domains, i.e., the higher the score the more ambiguous the word can be considered. We refer to this score as *ambiguity score*.

Specifically, the ambiguity score is a rank-weighted sum of the *variance* of the similarity values associated to any word appearing in the similarity lists[4]. If a word does not appear in a certain similarity list (e.g., *experiment* appears

[4] When n = 2, the variance is equivalent to the mean squared error.

```
AMBIGUITY-RANK(T_D, M, h)
 1  A_D, L_i, S_i, U, R, V, σ ← ∅
 2  for t ∈ T_D
 3       do
 4          for M_i ∈ M
 5               do L_i[t], S_i[t] ← MOST-SIMILAR(t, M_i, h)
 6
 7          U[t] ← L_1[t] ∪ ... ∪ L_n[t]
 8
 9          for w ∈ U[t]
10               do
11                  R[t][w] ← BEST-RANK(L_1[t], ..., L_n[t])
12
13                  σ[t][w] ← VAR(S_1[t][w], ..., S_n[t][w])
14
15          V[t] ← Σ_{w∈U[t]} σ[t][w] / R[t][w]
16
17  A_D ← SORT(T_D, V)
18  return A_D
```

Algorithm 2: Estimation of cross-domain ambiguity.

in the similarity list for *mouse* in the medical domain, but it does not appear in the CS one) the word is assigned a similarity value of zero for such list. Intuitively, if the lists share a large amount of words, and each of these words has close similarity values across the lists, then the dominant shared term can be assumed to have a consistent meaning in the different domains, and we consider it as not likely to be ambiguous. Instead, if the lists share few words, the dominant shared term is more likely to be ambiguous. The contribution of each word to the ambiguity score is weighted by the highest rank the word has across the similarity lists. The intuition is that, given a dominant shared term, its most similar words according to the language models should weight more in determining its ambiguity score.

The algorithm to measure the ambiguity score for each dominant shared term, and to produce the final ranking, is reported as Algorithm 2.

The algorithm takes as input the set of dominant shared terms $T_{\mathcal{D}}$, the language models $\mathcal{M}$, and a parameter h, which is the preferred length of each similarity list. For each term $t \in T_{\mathcal{D}}$, it computes an ambiguity score. This is done by first identifying the similarity lists ($L_i[t]$), together with the similarity values ($S_i[t]$), for each language model. Then, the union $U[t]$ between the lists is computed. The best rank ($R[t][w]$) of each word w in $U[t]$ across the lists is found, considering the first position (highest similarity) as the best one. If a word does not appear in a similarity list, BEST-RANK assumes that its rank in that list is $h + 1$. Finally, for each word in the union, the variance of its similarity values $S_i[t][w]$ across the different lists is computed by means of the VAR procedure and it is then weighted by dividing it by the best rank value. The sum $V[t]$ of the rank-weighted variances determines the ambiguity score.

The ranked list $A_{\mathcal{D}}$ of potentially ambiguous terms is produced, by sorting the terms in $T_{\mathcal{D}}$ by their ambiguity score.

4 Evaluation

To provide an evaluation of the proposed approach, we perform a set of experiments in which the results automatically produced are compared with those obtained on the basis of manually annotated data. In the following, we summarise research questions, research design in terms of data collection and analysis processes, as well as the validity procedure adopted.

4.1 Research Questions

The overall objective of the evaluation is to assess the effectiveness of the proposed method for the task of cross-domain ambiguity ranking. Furthermore, we want to understand how much the results obtained depend on the annotators involved in the definition of the ground truth against which the method is compared. Four main research questions (RQs) are defined to address these goals.

- **RQ1:** *What is the accuracy of the method in producing a list of terms ranked by cross-domain ambiguity?* With this question we want to provide a quantitative evaluation of the effectiveness of the proposed approach in ranking terms by cross-domain ambiguity, when compared with a ground-truth defined by manual annotators.
- **RQ2:** *To which extent different human subjects agree on the cross-domain ambiguity of terms?* The question aims to give a quantitative estimate of the agreement between different annotators in evaluating the degree of ambiguity of a term when this is used in different domain contexts.
- **RQ3:** *Which are the cases of inaccuracy?* The observed cases of inaccuracy of the method shall be classified in order to understand if and how the method can be further improved.
- **RQ4:** *Which are the cases of disagreement?* The observed cases of disagreement among annotators shall be classified to understand the limitations of the implemented validation approach.

4.2 Research Design

An overview of the applied research design is presented in Fig. 2. The design is structured into Data Collection and Data Analysis phases.

In the first phase, a set of fictional, yet realistic, elicitation Scenarios involving combinations of domains are defined. For each scenario, the method described in Sect. 3 (**Cross-domain Ambiguity** in Fig. 2) is applied to generate an ambiguity ranking, i.e., a list of terms ranked by cross-domain ambiguity. Then, data for evaluation are produced (**Evaluation Data Generation**).

These data consists of a sample ranking, extracted from the ambiguity ranking, and a list of sentence set, extracted from the different domain documents. These sentences include terms from the sample ranking, and are used to assess the variation of meaning of a term in the different domains. To this end, the sentence sets are manually annotated (**Manual Annotation**). This task consists in manually evaluating whether a term is used with different meanings in sentences belonging to different domains. This manual process produces scored sentence sets, i.e., sets of sentences to which a score is given based on the degree of variation of meaning of the considered term in the sentences. These sets are used to produce a ground-truth ranking (**Ground-truth Generation**) that will be compared with the sample ranking.

The Data Analysis phase is made of three main tasks. A first **Quantitative Evaluation** task in which the automatically generated sample ranking and the ground-truth ranking are compared by means of a ranking measure. With this measure, we aim to answer RQ1. In the same task, a measure of agreement is produced to estimate to which extent different annotators agree on the scores produced in the Manual Annotation task (RQ2). A **Qualitative Evaluation of Inaccuracies** is then performed to answer RQ3, as well as a **Qualitative Analysis of Disagreement** to answer RQ4, based on the measures obtained through the Quantitative Analysis and based on inspection of the sentences in the scored sentence sets.

The tasks that are depicted with double lines in Fig. 2, as well as the produced output, are replicated by means of two iterations. In a first iteration, the annotators are the two authors. In a second iteration, the annotators are workers recruited through Amazon Mechanical Turk (MTurk). The two iterations were performed for validity purposes (see Sect. 4.5).

In the following sections, the different tasks associated to the Data Collection and Analysis phases are described in details.

4.3 Data Collection

To provide the data to evaluate the proposed approach, we define a set of potential product development scenarios involving multiple domain experts, and we compute the ranked lists of ambiguous terms for the scenarios. Then, to have a ground-truth against which we can compare the automatically produced list, we generate ranked lists of ambiguous terms based on manual annotations.

4.3.1 Scenarios

The scenarios involve five domains, namely Computer Science (CS) Electronic Engineering (EEN), Mechanical Engineering (MEN), Medicine (MED) and Sport (SPO). Each scenario considers a subset of the domains. The first four can be regarded as interviews between an analyst with background in CS and a domain expert. The remaining three scenarios are group elicitation meetings, involving combinations of domain experts. The scenarios are briefly described

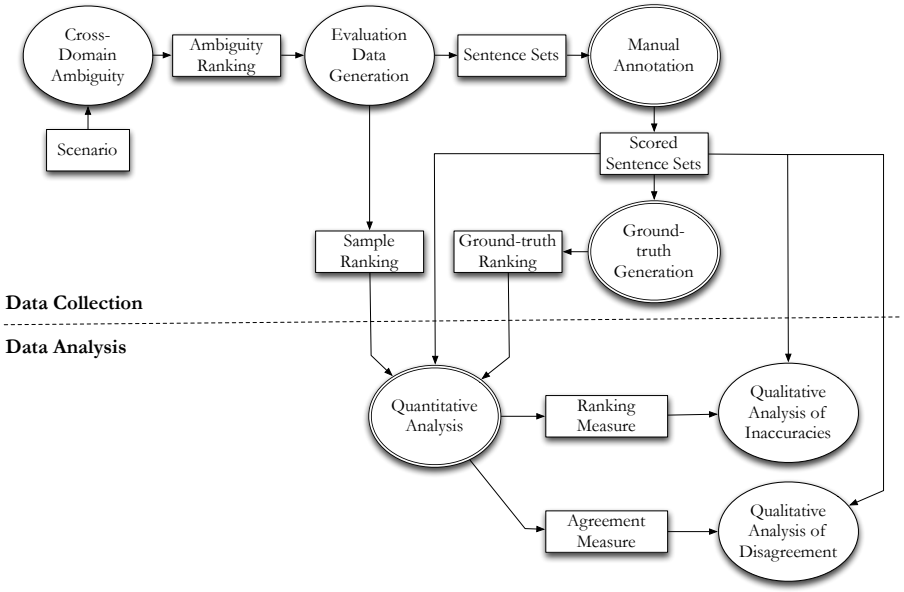

Fig. 2: Research design applied to evaluate the proposed method.

below, together with the acronyms of the domains considered. The descriptions have solely a narrative function, to suggest realistic contexts in which our approach can be used.

1. **Light Controller** [CS, EEN]: a generic software embedded in a controller to regulate the illumination of a room.
2. **Mechanical CAD** [CS, MEN]: a software to support the design of mechanical components.
3. **Medical Software** [CS, MED]: a software that supports the prediction of certain diseases, based on symptoms.
4. **Athletes Network** [CS, SPO]: a social network for athletes.
5. **Medical Device** [CS, EEN, MED]: a medical device for monitoring the hearth-rate of the patient, and linked to a mobile app.
6. **Medical Robot** [CS, EEN, MEN, MED]: a computer-controlled robot to perform surgery.
7. **Sport Rehab Machine** [CS, EEN, MEN, MED, SPO]: a high-tech machine for rehab, specifically oriented to athletes.

4.3.2 Cross-domain Ambiguity

The approach described in Sect. 3 is applied for each scenario. The approach was implemented in Python, using a set of support libraries, namely the

Table 1: Wikipedia pages for each domain.

Domain	Pages	Words	Vocabulary
Computer Science (CS)	10,000	3,985,740	104,907
Electronic Engineering (EEN)	8,568	4,576,917	100,272
Mechanical Engineering (MEN)	7,267	4,459,961	95,466
Medicine (MED)	10,000	5,470,284	150,617
Sports (SPO)	10,000	5,725,688	165,814

Python API for Wikipedia[5], `gensim`[6] for the `word2vec` implementation, and `spaCy`[7] for NLP support tasks. The source code of the algorithms, which can be used to replicate our experiments, can be downloaded from our GIT repository[8]. The repository includes also the generated language models, as well as all the data used in the evaluation.

In the application of the approach, we downloaded a maximum of 10,000 Wikipedia pages from each domain-specific Wikipedia portal involved in the scenarios (CS, EEN, MEN, MED, SPO). From our previous experiments (Ferrari et al. 2017b), this was regarded as a sufficient number of documents to learn the language models to be used in our context, considering the parameters' settings listed in Sect. 3.2. Table 1 reports the number of Wikipedia pages downloaded from each domain, together with the number of words, and the vocabulary size. Some of the domain-specific corpora (i.e., EEN and MEN) include less than 10,000 documents, since we crawled the entire set of pages included in the associated Wikipedia portals before reaching the threshold.

Given the domain-specific corpora, five different language models are generated. Then, the procedures Select-Terms and Ambiguity-Rank are applied for each scenario. For our scenarios, we set $k = 800$, i.e., the number of dominant shared terms that we wish to rank, $\rho = 0.3$, i.e., the frequency ratio, and $h = 100$, i.e., the length of the similarity lists that we compare with the Ambiguity-Rank procedure. These parameters' values were selected by the authors through a set of informal trials. With these settings, we produce the lists of dominant shared terms, ranked by their ambiguity score in the context of each scenario.

4.3.3 Evaluation Data Generation

To check that terms that are ranked higher have a diverse meanings in the domains involved, we perform an assessment using sentences from the Wikipedia pages of the domain-specific portals. These sentences are employed to check the actual in-context usage of a certain term in a given domain, and to compare its meaning when it is used in other sentences from other domains. Specifically, given a ranking, we first define a ranking sample of 20 terms, which includes

[5] `https://pypi.org/project/wikipedia/`

[6] `https://radimrehurek.com/gensim/`

[7] `https://spacy.io`

[8] `https://doi.org/10.5281/zenodo.1476902`

the top-10 terms from the ranking (set H, assumed to be made of ambiguous terms), followed by 10 additional terms randomly selected from the remaining ones (set L, assumed to be made of terms with lower degree of ambiguity). The selected structure of the sample is driven by the rationale that, in a realistic scenario, we want the majority of the ambiguous terms to be placed at the top of the ranking, while we expect that terms appearing in lower positions to be non-ambiguous to a comparable extent.

For each term of the ranking sample, we randomly create three sentence sets made of n sentences, one for each domain involved in the scenario. The sentences are randomly selected from the pages of the Wikipedia portals. Therefore, given the 20 terms of the ranking sample, we have 60 sentence sets for each scenario. Each sentence set includes from two to five sentences, depending on the number of domains involved. Considering the seven scenarios, our evaluation data is composed of 420 sentence sets, and 1200 sentences in total.

4.3.4 Manual Annotation

Human annotators are asked to evaluate whether the term associated to a sentence set is used with the same meaning in all the sentences of the set. The manual annotation step is performed in two iterations. In a first iteration, the authors of the current paper independently annotate the sentence sets. In a second iteration, a Human Intelligence Task (HIT) is performed on Amazon Mechanical Turk (MTurk), and two different annotators (i.e., *workers* in MTurk terminology) are required to annotate each set. An annotator works a single scenario at a time. The order of presentation of the sentence sets in a scenario to the annotator is random.

In both iterations, annotators are given a term together with an associated sentence set, and are asked: "Is the term used with the same meaning in ALL the following sentences?". The annotators provide their answer according to a four-point Likert Scale, in which 1 = *Exactly the same*; 2 = *Almost the same*; 3 = *Somewhat different*; 4 = *Extremely different*.

Examples were defined for each scenario, to give the annotators some guidance on how to perform the evaluation. Examples consists of a term and a sentence set for each of the four grades of the scale. Examples were carefully selected by the authors based on their experience in a preliminary evaluation[9]. For the Medical Software scenario, the examples used in the instrucions are reported below. The complete set of instructions and examples for each scenario are reported in our repository.

Annotation: *Exactly the same*
Term: room

– Sentence A: Finding **room** for the KY-28 in fighter aircraft was difficult.

[9] In this preliminary task, the authors annotated a group of 60 sentences for the Medical Software (CS, MED) and 60 sentences for the Medical Robot Scenario (CS, EEN, MEN, MED) – the sentence sets were not included in the evaluation presented in this work. The annotation was followed by a discussion on the annotated data and evident cases of disagreement, which allowed the authors to have an initial common ground to select the examples.

– Sentence B: In naval architecture, stowage is the amount of **room** available for stowing materials aboard a ship, tank or an airplane.

Annotation: *Almost the same*
Term: transfer

– Sentence A: Topics covered include knowledge assets and management, new product introduction and **transfer** of technology.
– Sentence B: Knowledge **transfer** is the practical problem of transferring knowledge from one part of the organization to another.

Annotation: *Somewhat different*
Term: reduction

– Sentence A: Her research takes a statistical approach to kernel methods, dimensionality **reduction**, and regularization in machine learning.
– Sentence B: A **reduction** drive is a mechanical device to shift rotational speed.

Annotation: *Extremely different*
Term: expression

– Sentence A: Modern devices may potentially allow computer related devices to respond to an individuals hand gesture, specific movement or facial **expression**.
– Sentence B: This **expression** can be further integrated to obtain the position r of the particle as a function of time.

4.3.5 Ground-truth Rank Generation

For each sentence set σ_t^i, the ambiguity score of the set is computed as the mean value between the answers of the two annotators. Then, for each term, the manual ambiguity score $N[t]$ of the term is computed as the mean between the ambiguity scores of the sets. More formally, given a term t, its $N[t]$ is computed as:

$$N[t] = \frac{\sum_{i=1}^{3} \sum_{j=1}^{2} ann_j(\sigma_t^i)}{6}$$

where $ann_j(\sigma_t^i)$ is the judgment given by the $j-th$ annotator to the σ_t^i sentence set. A ground-truth ranking of the terms is finally produced, by ordering them by their $N[t]$ value.

4.4 Data Analysis

4.4.1 Quantitative Analysis

Ranking Measure The first goal of this task is to evaluate the similarity between the automatically generated rank (sample ranking) and the manually

produced one (ground-truth ranking). The former is actually a ranking based on two sets, H and L, where: (a) all elements in H are considered ties, as well as all the elements in L; (b) all the elements in H are considered to be ranked higher than all the elements in L. The latter is a ranking that is based on the values of $N[t]$, and it can include ties too.

Note that the ties in the two rankings should have a different impact on the evaluation. A tie in the ground-truth ranking indicates that the annotators have observed a similar degree of ambiguity for two terms. In this case, the relative order in which the two terms appear in the sample ranking should not be relevant. For this reason, pairs of terms that are ties in the ground-truth ranking should be *excluded* from the evaluation. Once ties in the ground-truth ranking are excluded from the evaluation, any tie in the sample ranking should instead be considered an error from the method, because it has not taken a ranking decision in a case for which a ranking decision exists in the ground-truth.

This asymmetrical scenario is similar to the one of translation ranking in machine translation (Avramidis 2013), in which translation candidates are ranked, possibly with ties, by their semantic closeness to the original sentence by both a human expert, which defines the ground-truth, and by and automatic system. We thus adopt the Kendall's Tau (τ) formula with penalization of ties proposed for this task by Avramidis (2013). It is a variant of the Kendall's Tau measure (Agresti 2010; Kendall 1945) for the evaluation of ranked data with ties that also takes into account the two roles of ground-truth and prediction as explained above.

The τ formula of Avramidis (2013) ignores the ties that may occur in the ground-truth ranking, and counts solely the contributions given by the terms that have different manual ambiguity scores. Defined in terms of our notation, the formula is:

$$\tau = \frac{|(l,h) \in L \times H : N[l] < N[h]|}{|(l,h) \in L \times H : N[l] \neq N[h]|} \tag{1}$$

The measure counts the number of elements from the sample ranking whose order matches the one in the ground-truth ranking ($N[l] < N[h]$). Then, the measure divides the sum by the number of elements that are not ties in the ground-truth ranking, i.e, for which $N[l] \neq N[h]$. The value of the measure increases when more terms in the sample ranking are in the same order as in the ground-truth ranking. Note that the measure is not defined in the case in which all the elements in the ground-truth ranking are ties. Indeed, in this case, the ground-truth ranking is non-informative with respect to the task and no useful evaluation can be made.

The measure takes rational values in [0,1], where 0 indicates inverse correlation between the ranks and 1 indicates perfect correlation. A random predictor achieves $\tau = 0.5$, so, in our evaluation, our goal is to outperform this predictor with $\tau > 0.5$.

Agreement Measure The second goal of the Quantitative Analysis task is to estimate the agreement between annotators on the same sentence set. To this end, the Cohen's Kappa (Landis and Koch 1977) is used, and referred as κ. This measure takes rational values in [-1,1] where the following qualitative estimates are associated to the different ranges of the Cohen's Kappa: $\kappa < 0$, no agreement; $0 \leq \kappa \leq 0.20$, slight; $0.21 \leq \kappa \leq 0.40$, fair; $0.41 \leq \kappa \leq 0.60$, moderate; $0.61 \leq \kappa \leq 0.80$ substantial; and $0.81 \leq \kappa \leq 1$ almost perfect agreement.

The Cohen's Kappa is evaluated on the results of the annotators by considering the four annotation classes described in Sect. 4.3.4. We refer to this value as κ. Furthermore, given that low values of κ were observed, we also provided a second, permissive, estimate of the Cohen's Kappa, referred as κ_p. This estimate considers two annotations as the same answer if they belong to adjacent classes – i.e., if the answers differ solely by one unit as *Extremely different* and *Somewhat different*, or *Almost the same* and *Somewhat different*. To provide an intuitive estimate of the degree of agreement, we also provide the percentages of matches between annotations, both in the case of four classes (indicated as "%"), and in the permissive case ("$\%_p$").

It should be noted that the agreement is measured with the Cohen's Kappa solely within each iteration. Indeed, the agreement across iterations can be indirectly evaluated from a quantitative point of view by looking at the discrepancies between the τ obtained in the different iterations.

4.4.2 Qualitative Analysis of Inaccuracies

This task aims at providing an explanation for ranking inaccuracies, which can be of two types: (a) elements that are ranked lower in the ground-truth ranking, and that appear in H in the sample ranking; (b) elements that are ranked higher in the ground-truth ranking, and that appear in L in the sample ranking. With some abuse of terminology, we refer to the former as *false positives*, and to the latter as *false negatives*. Although we are not dealing with the evaluation of a classification problem, but with a *ranking* problem, we argue that this terminology provides the right intuition to explain the inaccuracies. To perform this analysis, the authors inspected the sample ranking and the ground-truth rankings of the first and second iteration. They considered solely those cases for which there was a difference between the sample ranking and *both* the ground-truth rankings – from the first and second iteration. The idea behind this choice is to analyse inaccuracies that do not depend on sources of disagreement between the authors and MTurk workers. Whenever a difference was observed for a certain term, the authors checked (1) its list of most similar words for the domain involved, and (2) the sample sets associated to the term, to identify a reasonable justification for the inaccuracy of the approach.

4.4.3 Qualitative Analysis of Disagreement

With this task, we evaluate which are the main categories of cases for which the annotators disagree in their assessment of the sentence sets. To this end, the authors informally inspect the cases of disagreement. The evaluation of agreement performed during the Quantitative Analysis showed a high level of agreement *within* each iteration, when measured as κ_p. Therefore, this analysis focuses on observing the disagreements *between* the two iterations, i.e., between the authors and MTurk workers. This analysis is performed by considering the two ground truth rankings produced in the two iterations, and observing relevant discrepancies in relation to the scored sentence sets.

4.5 Validity Procedure

The validity procedure adopted aims to ensure the validity of the data used in the study, and reported in this paper.

To ensure the validity of the annotations performed on the sentences during the Manual Annotation task, the annotation process is independently performed by two subjects. The inter-rater agreement is computed by means of the Cohen's Kappa (Landis and Koch 1977), and the scores used to produce the Ground-truth Ranking are based on (a) the averages of the scores given by the different annotators, and (b) the averages of the scores obtained by three different sentence sets including the same term. This is a form of triangulation that ensures that the manual ambiguity score for a term takes into account different viewpoints and different contexts in which the term may be used.

Second, we ensure the validity of the quantitative results reported by replicating part of the study by means of two iterations. Discrepancies of the results were evaluated and root causes of the discrepancies were assessed.

Third, to limit the researcher bias that may have affected the Manual Annotation task in the first iteration, in which the authors were involved, only the first author was exposed to the sample ranking – this could not be avoided due to technical reasons. Furthermore, in both iterations, the sentence sets were provided to the annotators in a random order, so that no trace of the sample ranking could be retrieved.

4.6 Results

In this section, we first present and discuss the lists of terms ranked by their degree of cross-domain ambiguity as estimated by the proposed approach, and then we systematically answer to the research questions.

4.6.1 Cross-Domain Ambiguity Ranks

Tables 2, 3 and 4 report the lists of dominant shared terms for each scenario, ranked by ambiguity score. We report the top-20 and the bottom-20 in the

Table 2: List of dominant shared terms ranked by ambiguity scores for Light Controller and Mechanical CAD interview scenarios.

Light Controller		**Mechanical CAD**	
Term	*Score*	*Term*	*Score*
news	1.475026003	**hull**	1.654419222
formula	1.466022543	house	1.447376925
relation	1.452569077	**argument**	1.391507139
surface	1.428484762	bar	1.361833239
motor	1.406120466	option	1.339622756
flash	1.405892082	room	1.33657024
studio	1.375377359	disk	1.328546185
contact	1.375058746	expression	1.317302937
interpretation	1.343498966	interpretation	1.316122939
bell	1.343395426	reduction	1.314786356
reduction	1.292722699	respect	1.306615558
head	1.282602451	relation	1.295833225
deal	1.246001682	representation	1.286481163
link	1.199083263	**formula**	1.270888584
ion	1.178597301	institute	1.24514362
desktop	1.171765465	port	1.241405669
pair	1.171018571	rest	1.22914137
profile	1.141438836	**statement**	1.215488445
particle	1.139154928	string	1.214932288
...	...	...	...
school	0.238366155	october	0.385298323
performance	0.23593757	state	0.366343161
term	0.233936501	category	0.360713789
article	0.22615044	december	0.360142567
september	0.225291357	period	0.359295452
conference	0.222977661	hour	0.355321192
number	0.221349352	cost	0.354388678
example	0.219640013	test	0.352514393
computer	0.216365358	space	0.346210712
range	0.214473014	advantage	0.345200372
student	0.21289965	september	0.343362203
march	0.210152016	day	0.33848604
system	0.203739261	minute	0.331166026
december	0.202965848	**time**	0.316961143
variety	0.20147449	market	0.31671382
point	0.2005448	range	0.289602617
science	0.197188439	**variety**	0.270675091
april	0.196663211	term	0.260514331
october	0.176143331	year	0.245205958
june	0.159893562	example	0.220515392

ranked lists. High ranks indicate higher chance of ambiguity. Here, we discuss notable cases, based on the different scenarios.

High-score Terms The term *argument* occurs among the most ambiguous terms in almost all the considered scenarios – the only exceptions are Light Controller and Athletes Network. If we check the most similar (i.e., related) words from the language models of e.g., CS and MED, we have the follow-

Table 3: List of dominant shared terms ranked by ambiguity scores for Medical Software and Athletes Network interview scenarios.

Medical Software		**Athletes Network**	
Term	*Score*	*Term*	*Score*
mouse	1.599703143	**michael**	1.849829619
matrix	1.542257577	protein	1.677702276
argument	1.478305447	**statement**	1.619878001
client	1.430145709	reduction	1.617916618
pair	1.4233354	loop	1.535914386
editor	1.419535955	string	1.522536711
arm	1.418809389	founder	1.503727767
strength	1.409200252	**formula**	1.489577327
house	1.396889719	washington	1.484234596
relation	1.369563912	effect	1.480257546
formula	1.35601545	edge	1.447221641
layer	1.348549133	mechanism	1.435133323
loop	1.340321432	layer	1.430993592
symbol	1.316503942	corner	1.424862031
reduction	1.316299964	threat	1.418862529
room	1.311479612	driver	1.418780755
statement	1.311244894	fire	1.412927231
expression	1.297684041	**surface**	1.411994676
surface	1.29639369	wave	1.411662759
...	...	...	...
report	0.383933955	category	0.503321974
concern	0.379293272	level	0.497828712
publication	0.346782484	sale	0.488006994
article	0.336694388	book	0.485710382
issue	0.333983175	term	0.481157591
history	0.330724481	company	0.456744991
student	0.321891081	market	0.448333875
award	0.319464903	april	0.409362809
time	0.308453773	history	0.401996814
december	0.299433864	child	0.394809601
april	0.283379154	june	0.377780407
october	0.279610472	september	0.365755703
march	0.279493799	student	0.355196116
september	0.27204809	**time**	0.353464531
june	0.264172934	award	0.350350368
category	0.263526951	article	0.343666994
term	0.2146701	march	0.341266415
variety	0.201947691	october	0.335467413
range	0.187092874	december	0.331697165
year	0.187065979	range	0.329518963

ing lists. CS: *proposition, truth, predicate, recursion.* MED: *arguing, opinion, critic, criticism.* In CS, an argument is intended as a logical argument, while in MED the term is mostly related to dispute. This is a case of lexical ambiguity, and, more specifically, a case of polysemy, in that the term *argument* has different, but related, meanings. Similarly, the term *statement*, appearing in several scenarios, may be a source of polysemy between, e.g., CS and and SPO domain experts, as the most similar words for these domains include the

Table 4: Lists of dominant shared terms ranked by ambiguity score for each meeting scenario.

Medical Device		**Medical Robot**		**Sport Rehab Machine**	
Term	*Score*	*Term*	*Score*	*Term*	*Score*
argument	2.023164853	**argument**	2.363125027	consequence	2.626065782
relation	1.921059422	respect	2.180597922	respect	2.596219047
formula	1.91556537	expression	2.180273772	**statement**	2.555115867
interpretation	1.904085589	consequence	2.173204542	**michael**	2.497249441
consequence	1.863527578	**statement**	2.094200528	story	2.398440676
expression	1.849218323	**ion**	2.089713398	**argument**	2.363125027
arm	1.838583297	father	1.952681603	**brother**	2.304480105
surface	1.811224084	**institution**	1.933082967	founder	2.238028413
house	1.802686127	relation	1.921059422	end	2.236240276
client	1.741765869	**formula**	1.91556537	ray	2.228692288
strength	1.730889913	interpretation	1.904085589	relation	2.216734358
mouse	1.716812815	career	1.890641217	stability	2.21369139
appearance	1.681019054	option	1.875132966	institution	2.201938176
ion	1.626415418	office	1.868301407	sense	2.199332125
statement	1.622403361	appearance	1.864991409	**surface**	2.193154903
discovery	1.615736883	man	1.863109403	robert	2.185128444
differential	1.615159841	compression	1.845858624	expression	2.180273772
sense	1.600477037	symbol	1.834537675	angle	2.164515441
gap	1.598580039	piece	1.823973036	option	2.157274619
segment	1.580185912	house	1.816749582	bill	2.147097884
...	...	...	...	...	...
range	0.288589908	keyboard	0.350738503	polygon	0.400052556
purpose	0.286876118	spin	0.350316254	organization	0.39887806
capability	0.284362316	architecture	0.338673159	processing	0.39547752
april	0.283379154	quantum	0.324911778	project	0.379329827
phone	0.282070409	instruction	0.322033875	battery	0.373586904
code	0.281997655	**time**	0.308453773	geometry	0.360184715
october	0.279610472	testing	0.30586906	keyboard	0.350738503
march	0.279493799	decrease	0.297651758	architecture	0.338673159
book	0.277109667	case	0.296671221	quantum	0.324911778
september	0.27204809	test	0.292459528	instruction	0.322033875
publication	0.266428293	electron	0.288590482	**time**	0.308453773
june	0.264172934	photon	0.284026579	test	0.292459528
group	0.243118643	phone	0.282070409	electron	0.288590482
school	0.238366155	code	0.281997655	photon	0.284026579
term	0.233936501	term	0.233936501	phone	0.282070409
article	0.22615044	conference	0.222977661	code	0.281997655
conference	0.222977661	computer	0.216365358	computer	0.216365358
computer	0.216365358	student	0.21289965	student	0.21289965
student	0.21289965	**variety**	0.20147449	**variety**	0.20147449
variety	0.20147449	century	0.179093637	century	0.179093637

following. CS: *clause, declaration, semicolon, goto*. SPO: *complaint, apology, announcement*. In CS, a *statement* is generally a source code instruction, while in SPO is a verbal communication, apparently with an apologetic flavour.

Another frequently occurring term in the ranks is *surface*. Its most similar words in the different domains include the following. CS: *shape, spherical, deformation*; EEN: *photoreceptor, anisotropic, gradient*; MEN: *adhesion, contact, smooth*; MED: *pore, layer, corneum*; SPO: *grass, asphalt, concrete*. These

words suggest that the term *surface* may be the source of a pragmatic ambiguity. Indeed, its meaning appears to depend on domain-specific viewpoints. In CS, a surface is an abstract concept, defined in terms of geometrical properties, while in EEN, optical physics concepts appear to dominate in the list of most similar words. In MEN, the term is mostly associated to physical properties, in MED to body parts, and in SPO a surface is the floor over which a certain sport is practiced. If, for example, we inspect one of the CS documents in which the term is used, we find the sentence: *Onscreen objects must be 'clipped' to the screen, regardless of whether their surfaces are actually visible*; instead, in MEN, we find: *Some transmission units also have Winter mode, where higher gear ratios are chosen to keep revs as low as possible while on slippery surfaces*, and in SPO we have sentences such as: *Surface variations can have a significant effect on how ground balls behave and are fielded as well as on baserunning.* These examples show how the term is used with different domain-specific flavours of meaning, and give an idea of the concepts triggered in the mind of stakeholders when hearing the term *surface*.

A final, interesting case, is the term *formula*, occurring in most of the scenarios. Looking at its related words, we have the following. CS: *quantifier, lemma, axiom, negation, propositional.* EEN: *summation, quadratic, logarithm.* MED: *elemental, pure, mixing, adding.* SPO: *f1, mclaren, mercedes, ferrari.* In the case of the Light Controller software, involving CS and EEN, the term may be regarded as a case of generality, in that for CS a *fomula* is normally a logical formula, while in EEN is a mathematical formula, typically of a continuous function. When also MED experts are involved, as in the Medical Device case, the term may be interpreted as a chemical formula. Finally, a lexical ambiguity case may occur in the Athletes Network scenario, in which a SPO expert may consider *formula* as the well-known type of road racing, as, e.g., *Formula 1.*

Low-score Terms The terms at the bottom of the lists in the tables are those that obtain the lowest ambiguity score. These terms include names of months (*december, april, october*, etc.), as well as well defined concepts such as *time*, *space*, *hour*, or generic terms such as *variety*, which can be adapted to different contexts without changing its meaning – its most similar words include *wide, range, diverse*, etc.

We also observe that the term *term* has a low ambiguity score across domains, and the same most similar words occur between domains. However, by looking at similar words, we have *meaning, terminology, word* but also *denominator, factorial*, etc. This indicates that the term is used with different meanings (linguistic vs mathematical term) within the each single domain. Additional strategies are required to account for these cases of in-domain ambiguity, as we also notice in the following sections.

4.6.2 RQ1: What is the accuracy of the method in producing a list of terms ranked by cross-domain ambiguity?

Table 5 reports the value of τ obtained for the different scenarios in the two iterations. We see that the average value of τ for interviews is above random guessing (i.e., above 0.5) in both iterations, with slightly lower performance for the second one (0.58 vs 0.64), and a peak of 0.81 for the Medical Software scenario in the first iteration. For meetings involving more than two domains, we still have an average τ above random for the first iteration (0.62), while poor performance are observed for the second iteration (0.36). The best performance are obtained for the Medical Device scenario in the first iteration (0.88). However, high discrepancy is observed in the results between the two iterations, and the potential sources of this outcome are discussed extensively in Sect. 4.6.5. Considering the τ values obtained in the different iterations, we can say that the method has acceptable accuracy for interviews and meetings with a limited number of domains, while it decreases its effectiveness with meetings involving several domains.

Table 5: Accuracy of the method in producing lists of terms ranked by cross-domain ambiguity.

Scenario	*τ (1st iteration)*	*τ (2nd iteration)*
Light Controller	0.39	0.56
Mechanical CAD	0.75	0.59
Medical Software	**0.81**	0.45
Athletes Network	0.60	0.72
Average (interviews)	**0.64**	**0.58**
Medical Device	**0.88**	0.48
Medical Robot	0.57	0.47
Sport Rehab Machine	0.42	0.17
Average (meetings)	**0.62**	**0.37**
Average	0.63	0.49

4.6.3 RQ2: To which extent different human subjects agree on the cross-domain ambiguity of terms?

Table 6 reports the results for the κ agreement, and its permissive κ_p variant. We see that the average κ, measured on four classes, is low for both iterations, with $\kappa = 0.38$ (fair, 54% of the sentence sets) and $\kappa = 0.09$ (slight, 29%) for the second one. This low value of agreement, especially for the second iteration, would not allow to draw any conclusion about the actual effectiveness of our method, since it would imply that the ground truth used for evaluation is ill defined. However, we considered that adjacent classes (e.g., "Extremely different" and "Somewhat different") may be treated as equivalent, taking into account the complexity and high subjectivity of the annotation task. Therefore, we introduced the κ_p measure, which, we recall, considers as agreement

Table 6: Agreement between annotators in evaluating the variation of meaning of dominant shared terms.

Scenario	*1st iteration*				*2nd iteration*			
	κ	%	κ_p	$\%_p$	κ	%	κ_p	$\%_p$
Light Controller	0.26	43%	0.65	82%	0.09	37%	0.93	97%
Mechanical CAD	0.36	57%	0.77	88%	0.09	26%	0.30	59%
Medical Software	0.53	65%	1.00	100%	0.05	30%	0.90	95%
Athletes Network	0.45	62%	0.77	88%	0.07	25%	0.65	82%
Medical Device	0.39	53%	0.90	95%	0.10	27%	0.47	71%
Medical Robot	0.29	45%	0.80	90%	0.17	35%	0.58	78%
Sport Rehab Machine	0.38	53%	0.90	95%	0.04	20%	0.52	75%
Average	0.38	54%	0.83	91%	0.09	29%	0.62	80%

all the answers that differ solely by one unit (i.e., if the answers belong to adjacent classes). When using this measure, we see that annotators agree on 91% and 80% of the cases in the first and second iteration, respectively, leading to an agreement of $\kappa_p = 0.83$ (first iteration, almost perfect) $\kappa_p = 0.63$ (second iteration, substantial). These values of agreement indicate that the ground truth rankings are internally consistent. Furthermore, they show that within each iteration human subjects agreed on the cross-domain ambiguity of the terms, estimated as variation of meaning of the terms when used in different domain-specific contexts. This implies that, when human subjects are provided common instructions, they tend to agree on the cross-domain ambiguity of terms.

4.6.4 RQ3: Which are the cases of inaccuracy?

False Positives We consider false positives all those terms that are ranked high in the sample ranking, while they are ranked low in both ground-truth rankings from the two iterations. These terms fall into the following categories:

Non-representative Samples In some cases, the samples used to provide the sample sets appear not to be representative of the most common domain-specific meaning of a certain term. For example, the term *editor*, which is in the top-10 ambigous terms for the Medical Software scenario (see Table 4), is considered non ambiguous by all annotators in most of the samples, in which the term is used to indicate the person responsible for the editorial aspects of a publication, as in the following sentence: *He has been an associate editor of the ACM Transactions on Graphics*. However, by looking at the most similar words in CS we have *tex, wysiwyg, emacs, tux*, etc., while the most similar in MED are *editorial, bulletin, authored, blog*. Therefore, we argue that in CS an *editor* is generally a computer program, although the sampled sentences are not representative for this meaning. In a sense, these cases can be considered as *true* positives rather than false positives.

High-level Abstract Concepts Some terms indicating high-level abstract concepts, such as *consequence*, *cause*, *idea*, and the term *concept* itself do

not appear to change their meaning, regardless of the context in which they are used. For example, the term *consequence*, which is in the top-10 for the Medical Device scenario (see Table 4) is used with about the same meaning in the sentences of the involved domains. For example, in CS we find: *As a consequence, Symbian's market share fell, and application developers for Symbian dropped out rapidly.* In EEN: *This is an unavoidable consequence of any system that samples an otherwise continuous signal at discrete intervals or locations.* In MED: *He believed that this result was a consequence of the flow of subtle energy between himself and the patient.* In all the three sentences, a *consequence* is the effect of some phenomenon. However, the most similar words in the three domains are different, and are as follows. CS: *unintended, circumstance, imposed, situation*; EEN: *invariance, existence, arises, chaotic*; MED: *detrimental, vulnerability, stressor, implication.* Overall, the three words lists of 200 words associated to the term *consequence* in the considered domains have solely 9 words in common. We therefore conclude that some terms indicating high-level abstract concepts, although generally accompanied by different words in different domains, do not change their meaning. Our approach is not currently able to automatically discard these terms from the ambiguous ones.

Undetected Domain-specific Usage Some terms have been noticed to actually have domain-specific usages, although the annotators, which are not experts in the domains, did not notice the different shades of meaning in the example sentences. This is the case of the term *ion*, in the Medical Robot scenario (Table 4). Consider the following sentences from two of the four domains involved. EEN: *A beryllium ion has been trapped in a superposed state.* MED: *The cellular mechanisms of commotio cordis are [...] probably related to the activation of mechanosensitive proteins, ion channels.* The large majority of the annotators considered the meaning of *ion* as the same in the reported sentences. The following are items extracted from the lists of most similar words. EEN: *ionization, implantation, donor, energetic, bombardment.* MED: *hydrogen, atp, intracellular, peroxide.* This suggests that in EEN the term *ion* is mostly related to electronics aspects, while in MED is used in relation to biological aspects. It can therefore be a source of pragmatic ambiguity. This shade of meaning could not be noticed by our domain-ignorant annotators, and the authors admit to have had a superficial understanding of the presented technical sentences. To better address these cases, our evaluation should have involved domain experts.

Multiple Factors Some unambiguous terms are ranked high due to the combination of multiple factors, as in the case of the term *brother* in the Sport Rehab Machine scenario (Table 4). For this specific case, two factors brought this term to the top-10: (a) the encyclopedic nature of the text in Wikipedia, which speaks about biographies and relatives, and therefore includes the term *brother* as a frequent term – while the term may not be used so frequently in the technical languages of the domain; (b) the nature of the term *brother*, often accompanied by proper names, which widely vary across domains; (c) the unbalanced number of occurrences of

the term between different domains – the term occurs 1139 times in SPO and 344 in MEN, slightly above the 0.3 frequency ratio threshold. These three factors altogether lead to the following lists of most similar terms for the two domains. SPO: *uncle, father, husband, son, sister*; MEN: *stanley, son, samuel, matthew, oliver*. While in SPO the term occurs sufficiently often for `word2vec` to "learn" its actual meaning of close family relative, in MEN the term is associated with names of *brothers*. We argue that, to address these corner cases, different sources of domain knowledge other than Wikipedia should be used to train the `word2vec` algorithm.

False Negatives We consider false negatives all those terms that are ranked low in the sample ranking, while they are ranked high in both ground-truth rankings. These terms fall into the following categories:

Non-representative Samples As for false positive cases, some of the randomly selected samples were not actually representative of the usage of a term in the domains. This is the case of the term *capacity*, for the Light Controller scenario, for which we have the following sentences in the sample set. CS: *With the Internet a new capacity for collective, including planetary-scale, problem solving was created.* EEN: *It also took some time for other storage manufacturers to release products: most were smaller devices aimed at the professional market, and focussed on speed rather than high capacity.* The term is clearly used with different meanings in the example sentences. However, the lists of 200 most similar words in the two domains overlap by 71 elements, including *rate, sram, throughput, rate*, etc. Therefore, we argue that, in the two domains, the term is actually used with similar meanings (i.e., in general associated with *memory capacity*) but the selected samples may not be representative of the actual usage.

In-domain Ambiguous Terms Some of the terms are frequently used with different meanings within the same domain, and can be therefore considered ambiguous within the domains. This is the case of a term such as *law*, which was considered ambiguous by the annotators in the Light Controller scenario. This term is used in EEN in sentences such as *The capacitance can be calculated in a straightforward way using Gauss's law*, but also *Wireless security cameras are also used by law enforcement agencies to deter crimes*. Similarly, in CS we find *The interpolation is essentially a power law*, and also *The platform is provided by CitizenGlobal, free of charge, to law enforcement agencies*. The physical, mathematical and juridical senses of *law* co-exist in the considered domains, and common words between the lists of most similar terms include *governed, obligation, evidence, investigation* but also *thermodynamics, conservative, peirce*. These peculiar cases cannot be detected with the current approach.

4.6.5 RQ4: Which are the cases of disagreement?

We have observed that, considering a permissive agreement (see Sect. 4.6.3), the authors agree on 91% of the Sentence Sets, with an average $\kappa_p = 0.81$.

Similarly, MTurk workers agree on 80% of the cases, with average $\kappa_p = 0.63\%$. Since these agreements can be considered almost perfect and substantial, respectively, it is useful to focus on the sources of disagreement *between* the two groups of annotators, which led to differences in terms of ground-truth rankings and, in turn, in terms of performance. Below are the classes of cases that were observed.

Named Entities Named entities, such as person names, places and organisations, are often used in different linguistic contexts. For example, the name *Michael* appears in the top-10 list of the Sport Rehab Machine (Table 4), and it is the most ambiguous term for the Athletes Network (Table 3). The term occurs in typical encyclopedic sentences such as *The Foundation announced that Michael Howden [...] was appointed as its new CEO* (CS), or *[...] professionals like Michael Iaconelli [...] and Luke Clausen are household names to many American sports fans.* (SPO). The authors considered the usage of the term as the same, in that it was always representing a proper name. Instead, some MTurk workers considered the term to have different meanings, since it was referring to different subjects. Something similar occurred for the term *institution*, in the Medical Robot scenario (Table 4), in which the term is used to refer different actual institutions, as e.g., *Institution of Engineering and Technology (MIET)* (EEN), or *Smithsonian Institution* (MEN).

Absence of Context/Domain Knowledge We noticed that MTurk workers tended to provide different judgments with respect to the authors especially in presence of technical sentences, or sentences that required some domain knowledge to be understood. For example, the workers considered the term *arm*, from the Medical Software Scenario (Table 3) to have almost the same meaning in the following sentences: *For example, the OMAP3 processors include a ARM Cortex-A8 and C6000 DSP* (CS); *A small amount (0.1 ml) of diluted (1/50 MLD) diphtheria toxin is injected intradermally into one arm of the person and a heat inactivated toxin on the other as a control* (MED). The two sentences are actually hard to understand without prior domain knowledge, and MTurk workers may have provided a random judgment for this case. To prevent these cases, we should have required workers to give an estimate of their understanding of the sentences, so to evaluate the reliability of their judgment.

Incorrect Interpretation of Instructions For the multi-domain scenarios, we observed particularly low accuracy in the second iteration. We argue that this may be mainly due to incorrect interpretation of the instructions given by the authors to the MTurk workers. Looking at their answers for the Medical Device scenario (Table 4), we observed that the term *house* was considered to have almost the same meaning when presented the following set of sentences: *In 2012, reCAPTCHA began using photographs of house numbers taken from Google's Street View project* (CS); *Real objects can be divided into two abstractions: discrete objects (e.g., a house) and continuous fields [...]* (EEN); *In April 2014 U.S. House Democrats Henry*

Waxman [...] (MED). While in the first two sentences the term can be considered to have the same meaning, in the third sentence the meaning is different. We argue that some MTurk workers may have incorrectly interpreted the question *Is the term used with the same meaning in ALL the following sentences?* – which they were required to answer – as *Is the term used with the same meaning in SOME of the following sentences?* This problem may have had increasing impact on scenarios involving a higher number of domains, and corresponding sentences.

Low Reliability of (some) MTurk Annotators In some cases, we argue that MTurk annotators did not perform their tasks in a reliable manner, and provided random answers to complete their tasks faster and get the reward. One peculiar case is the term *hull*, which is the most ambiguous term in the Mechanical CAD scenario (Table 3), and was ranked as the most ambiguous also in the ground-truth ranking of the fist iteration. However, one of the annotators considered the term to have exactly the same meaning when presented with the following couple of sentences: *a pseudotriangulation of a point set is a partition of the convex hull of the points into pseudotriangles* (CS); *double bottoms [...] have two hull layers only in the bottom of the ship but not the sides* (MEN). In the opinion of the authors, the provided annotation could be explained only by considering that the specific MTurk worker provided a random answer – although the absence of domain knowledge may have contributed to the random choice.

4.7 Discussion

This section presents a general discussion in light of the answers to the RQs, together with some issues concerning the applicability of the presented approach.

- **Accuracy of the Approach:** with few exceptions, the approach outperformed a random predictor for the interview cases in both iterations. For the first iteration, average results above random guessing were obtained also for group meetings. Significantly low performance were observed for the Sport Rehab Machine scenario, involving five domains, in both iterations. From these results, and considering the several factors affecting them – see below –, we can state that the method is generally accurate for interview cases and meetings involving a limited number of domains, while it is less accurate for cases in which many domains are involved.
- **Factors Affecting Accuracy:** based on the qualitative evaluations performed, the presented quantitative results on the accuracy of the approach appear to depend on several factors, including: 1) the context and domain knowledge required to annotate the sentences, which were not always understood by authors and MTurk workers; 2) the source documents used, which include encyclopedic terminology, not representative of domain-specific terminology; 3) the sentence samples used, which were not always representative of the domain-specific meaning of terms; 4) the

difficulty in checking the reliability of MTurk workers. All these factors contributed to decreasing the accuracy of the presented approach. Further tuning of the method and associated experiments will be performed to reduce the influence of these factors.

- **Agreement:** we have observed that, within each iteration, a high degree of agreement on the cross-dependent ambiguity of terms was obtained. However, discrepancies in the judgments – qualitatively observed by comparing different ground-truth rankings – were obtained between the two iterations. Many of these discrepancies can be explained with incorrect interpretation of instructions by MTurk annotators, and their lack of domain knowledge and context. Overall, given these observations about the agreement, and despite the researcher bias, the authors are keen to consider the results of the 1st iteration as more reliable with respect to those obtained from the 2nd iteration.
- **Coarse-grained Domains:** the approach is in principle applicable for meetings involving stakeholders from different domains, and when each domain can be identified in a *coarse-grained* way, e.g., CS, EEN, MEN. In practice, people from highly specialised sub-domains may be involved in requirements elicitation meetings. Furthermore, stakeholders may have expertise in more than one domain, and different ambiguities may occur in these cases. These practical scenario variants are not considered in the current work, and, although the approach is in principle applicable to these contexts, different solutions or appropriate tuning may be needed.
- **Corpora Size and Availability:** we used corpora of about 10,000 documents. This number of documents resulted to be sufficient for our purposes, provided that the parameters for `word2vec` are those indicated in Sect. 3.2. In general, `word2vec` is more effective if more documents are used as input, and, therefore, we can infer that also the proposed approach may be more effective with a larger number of documents. However, one should consider that large amounts of domain-specific documents may be hard to retrieve. This lack of domain knowledge sources may set an inherent limitation of the proposed method, especially if one wishes to consider specialised sub-domains, for which a limited number of representative documents may be available.
- **Suitability of Wikipedia:** we adopted Wikipedia as source of domain knowledge in the form of documents. Wikipedia is an encyclopedia, and the included documents speak *about* domains, and are not fully representative of the domain-specific language that could be used by domain experts. Besides including encyclopedic/historical terms and information – as already noticed in Sect. 4.6.4 and 4.6.4 – Wikipedia documents may not include typical terms and concepts used within domain-related discussions. To address these issues, different sources of domain knowledge should be used, as, e.g., scientific articles or domain-specific manuals. As an alternative, encyclopedic/historical content should be excluded from the corpora used to build the language models.

- **Rare Terms:** our work focuses on dominant shared terms, i.e., terms that are frequent in different domains. Rare terms are harder to deal with, since they are less frequent in the input corpora, and therefore their semantic representation in the form of word embeddings is less informed, and in turn, less accurate. Therefore, comparison of embeddings of rare terms to identify ambiguity with the proposed approach is not recommended.
- **Ambiguity Score:** the actual representativeness of the values of the ambiguity score needs to be assessed. At this stage, we only consider the ambiguity ranking resulting from these scores. Therefore, while we can say that certain words are more likely to trigger ambiguity than others, we cannot say which score values indicate a higher likelihood of ambiguity.
- **Offline vs Online:** the method identifies a list of dominant shared terms based on domain documents, and does not consider the actual words that may be used during an elicitation meeting. In a sense, the approach is useful for offline meeting preparation. To take into account the specific language context of the elicitation meeting, and possibly perform online ambiguity checking, the approach should be extended with an iterative document crawling/language models generation process, triggered by the words used during the meeting. In a sense, the domains involved in the meeting should be inferred based on the words used by the stakeholders, representative documents shall be crawled, and language models should be built based on these documents. This is a radically different application scenario, which may require a non-trivial modification of the proposed approach.
- **Computational Cost:** the main computational cost of the method is related to the construction of the language models. Learning a `word2vec` model on the 10,000 documents we collected for each domain required about 15 minutes on a desktop machine. A language model for a domain has to be learned once, then it can be used multiple times against different domains. The Select-Terms and Ambiguity-Rank algorithms require few seconds to run.

4.8 Threats to Validity

Construct Validity The main threat to construct validity is due to the usage of indirect measures to identify the ambiguity of terms in requirements elicitation interviews and meetings in general. Specifically, to approximate the terminology of these meetings, we used the terms in domain-specific Wikipedia portals, rather than the terms used in real-world meetings. This threat could not be mitigated, due to the lack of NL data from requirements elicitation meetings.

To estimate the ambiguity of a term in the Manual Annotation task (see Sect. 4.3.4), we asked annotators to assess the variation of meaning of the term in different sentences. This is an *indirect* measure of ambiguity, and the outcome may depend on the specific sentences considered. Moreover, subjective assessments were leveraged to produce the quantitative measure used to compare manual and automated rankings. To mitigate these threats, three sentence

sets were used to assess the ambiguity of a term, as well as annotations from two distinct subjects for each set, and for each iteration. Average measures were used to mitigate subjectivity and context-dependency of the produced manual rankings. Furthermore, to create a common ground for assessment, before the first iteration the authors performed a preliminary evaluation (see Sect. 4.3.4) to define guidelines and examples for the actual Manual Annotation task.

Internal Validity The main threats to internal validity is related to the process used for the Manual Annotation task. Indeed, the annotations were performed *after* the generation of terms by the presented approach. Furthermore, the sample rank used as a source for the annotation task included solely a subset of the terms. Therefore, cross-domain ambiguous terms may exist, which did not emerge in our evaluation. Another threat is the researcher bias, which may have occurred in the 1st iteration. To mitigate this threat, corrective measures were taken, as explained in Sect. 4.5. Furthermore, the annotation was not performed by domain experts, and therefore they could fail in detecting ambiguities, as we also noticed in Sect. 4.6.4 and 4.6.5. Threats concerning domain knowledge of the annotators could not be entirely mitigated.

External Validity We experimented within a rather representative variety of domains as well as domain combinations. Therefore, we argue that, based on the evaluation, the method can be considered acceptably accurate for interviews between CS and other domains, as well as for meetings involving a limited number of domains. Its accuracy decreases for group meetings involving several domains (Sect. 4.6.2).

5 Related Works

As RE activities are typically NL-intensive, NLP techniques have been extensively applied in RE for several tasks covering a large part of the requirements process. These tasks include requirements tracing (Guo et al. 2017; Sultanov and Hayes 2013), categorisation (Casamayor et al. 2012), retrieval (Natt och Dag et al. 2005), glossary extraction (Gacitua et al. 2010; Quirchmayr et al. 2017), model synthesis (Robeer et al. 2016; Yue et al. 2015), app review analysis (Jha and Mahmoud 2017; Maalej and Nabil 2015), analysis of legal requirements (Evans et al. 2017; Sleimi et al. 2018), and detection of defects such as equivalent requirements (Falessi et al. 2013) and ambiguity (Berry and Kamsties 2004; Tjong and Berry 2013). An overview of the techniques and their positioning in the RE process is presented in our previous work (Ferrari et al. 2017a). Given the focus of our paper, in the following we discuss works from the RE field that specifically relate to defect detection in general and ambiguity in the specific. Then, we refer the most relevant works from the NLP field that treat the problem of word-sense disambiguation, which is connected to the topic of domain-dependent ambiguity addressed in this paper.

5.1 Defect Detection and Ambiguity in RE

Ambiguity in NL has been extensively studied in RE, especially in relation to its occurrence in written requirements. The works can be broadly partitioned into those oriented to ambiguity prevention, and those aimed at ambiguity detection, discussed in the following paragraphs. We also include a paragraph on works specifically oriented to pragmatic ambiguity, since this is among the main focuses of our paper.

Ambiguity Prevention In the first group of works, strategies were defined to prevent ambiguities by means of formal approaches (Ambriola and Gervasi 2006; Kof 2010; Mich 1996) or constrained natural languages (Arora et al. 2015; Mavin et al. 2009). Ambiguity prevention is among the main objectives of the work of Kof (2010), and of tools like Circe-Cico (Ambriola and Gervasi 2006) and LOLITA (Mich 1996), which transform requirements into formal/semi-formal models. By means of this computer-mediated model synthesis process, ambiguity in requirements are avoided or limited. Concerning the use of constrained natural languages, the EARS template (Mavin et al. 2009) and the Rupp's template (Pohl and Rupp 2011) are well known constrained formats for editing requirements, and improving their uniformity and clarity. Arora et al. (2015) defined an NLP approach to check the conformance of requirements to these templates, and experimented the approach in an industrial case study.

Our work is oriented to ambiguity prevention in requirements elicitation meetings. It differs from the corpus of literature on ambiguity prevention since it stresses the domain-dependent component of ambiguity, which is not considered by any of the mentioned works.

Ambiguity Detection Techniques for ambiguity detection are mainly *rule-based*, i.e., based on linguistic patterns to be matched within requirements (Berry et al. 2003). Automated tools such as QuARS (Lami et al. 2001), SREE (Tjong and Berry 2013) and others (Femmer et al. 2017; Gleich et al. 2010) were developed according to this philosophy. Interestingly, research on ambiguity detection in NL requirements with rule-based approaches has reached sufficient maturity to be applied in real-world industrial projects, as shown in the recent work of Femmer et al. (2017) and Ferrari et al. (2018b). Commercial defect-detection tools are also available, such as Qualicen Scout[10] and QVScribe[11].

Other works (Chantree et al. 2006; Yang et al. 2011) focus on the usage of *statistical* approaches to detect particular types of ambiguity cases, the so-called *innocuous ambiguities* – i.e., linguistic ambiguities that have one single reading in practice. Chantree et al. (2006) specifically focus on the detection of coordination ambiguities, which are related to the usage of "and" or "or" conjunctions. Yang et al. (2011) deal with the problem of anaphoric ambiguities, which depend on the potentially ambiguous interpretation of pronouns.

[10] `https://www.qualicen.de/en/`

[11] `https://qracorp.com`

Both rule-based and statistical approaches focus on lexical and syntactic ambiguities (see Sect. 2), although semantic and pragmatic ambiguities cases are detected at lexical level in some of the works, as, e.g., Gleich et al. (2010). Our work differs from the ones on ambiguity detection for its main goal, which is ambiguity prevention. Furthermore, we focus on domain-dependent ambiguities, including lexical, pragmatic and generality cases.

Pragmatic Ambiguity Pragmatic ambiguity received little attention from the RE community interested in NLP although contextual factors, and in particular the domain knowledge of the stakeholders, were observed to play a prominent role in the occurrence of ambiguity in requirements elicitation (Ferrari et al. 2016).

Among the few works on pragmatic ambiguity, Dalpiaz et al. (2018) recently proposed the usage of automatically generated visual models to pinpoint a specific type of ambiguity (i.e, near-synonymy), which may depend on stakeholders' viewpoints. Differently from us, the focus of Dalpiaz et al. (2018) is on ambiguities that may be triggered by the usage of different words to refer to the same concepts. On a different note, Rodriguez et al. (2018) focus on preventing pragmatic ambiguity between customers and developers, by facilitating the access to the domain knowledge available in Wikipedia. To this end, they developed an NLP tool called WIKINA (Wikipedia Knowledge Acquisition). Similarly, Lian et al. (2016) present MaRK (Mining Requirements Knowledge), a tool for retrieving information from domain documents, to facilitate the acquisition of domain knowledge. Our work differs from that of Rodriguez et al. (2018) and Lian et al. (2016), since they mainly focus on domain knowledge retrieval rather than linguistic ambiguity. The first author of the current paper also worked on pragmatic ambiguity with a technique that leverages the comparison of knowledge graphs generated from domain-specific Web-pages (Ferrari and Gnesi 2012). The current work continues this path of research. It mainly differs from the previous work since it uses state-of-the-art word embeddings technologies to compute the terms' meaning in different domain contexts, instead of more computationally expensive graph-based measures. Furthermore, notwithstanding its acknowledged limitations, the current approach has undergone a rigorous evaluation process.

5.2 Word-sense Disambiguation

Word-sense disambiguation (WSD) is a classical NLP task, which, given a term and its linguistic context, i.e., neighbouring words and sentences, aims to determine the meaning of the term among a pre-defined set of possible meanings (Agirre and Edmonds 2007; Navigli 2009). WSD can be regarded as a classification task, in which the item to classify is a term given its context, and the classes are its possible senses. The extensive list of techniques proposed in this field is broadly partitioned by Navigli (2009) into (1) supervised, (2) unsupervised and (3) knowledge-based approaches.

1. Supervised approaches use machine learning techniques, such as support vector machines (SVM) (Lee and Ng 2002) or Naive Bayes (Pedersen 2000). These techniques leverage manually labelled textual corpora in which, for each term, the *true* sense has been manually established.
2. Unsupervised approaches do not make use of labelled corpora, but are based on the idea that words with the same sense will have the same neighbouring words. These methods induce word senses by clustering terms that appear in the same contexts. Hence, a term may appear in different clusters, depending on how many senses it has. A survey on unsupervised methods is presented by Pedersen (2007).
3. Knowledge-based approaches leverage the information available in different dictionaries or thesauri, such as WordNet[12] (Miller 1998). These approaches identify the sense of a term in a text by taking into account its linguistic context in the text, and the relation with other terms (e.g., hypernyms, meronyms) as made explicit by the dictionaries.

More recently, semi-supervised approaches, which combine principles from (1) and (2) have also been developed, as, e.g., the technique presented by Taghipour and Ng (2015). The authors exploit pre-trained word embeddings to provide semantic information to a supervised classifier for WSD. Also knowledge-based (3) and unsupervised (2) approaches have been combined, e.g., by Chen et al. (2014), which use word embeddings together with WordNet to create vectors that represent the different senses of a word. Flekova and Gurevych (2016) make a step forward, and create joint word- and supersense-embedding models, in which the vector representing a word takes into account both its fine-grained sense, and its supersense as defined in WordNet – i.e., its class of meaning, as, e.g., animal, person, communication. A survey on existing work on vector-based representations of meaning has been published by Camacho-Collados and Pilehvar (2018). Finally, with the advancements in neural networks for NLP, also these techniques have been extensively used and adapted to the WSD context (Raganato et al. 2017; Yuan et al. 2016).

Our approach can be considered as *unsupervised*, since no labelled data are used to train it. However, our goal is not to disambiguate a term in a textual context, or searching a precise sense for it, as typical for WSD techniques. Instead, we aim at finding potentially ambiguous terms without having a textual context, but knowing solely the domains that will be involved in a certain communication. Our work also differs in spirit from the WSD literature: we do not assume that a pre-defined set of senses exists. Instead, we identify potential *shades* of senses, based on the usage of a term in domain-specific contexts.

6 Conclusion and Future Work

Ambiguity in natural language is a complex phenomenon that has been largely studied in requirements engineering. However, most previous applied work on

[12] `https://wordnet.princeton.edu`

the topic has focused on ambiguities that are triggered by domain-independent terms and constructions. In the current work, we propose one of the first approaches to identify domain-dependent ambiguities that may occur in requirements elicitation interviews and meetings involving stakeholders from different domains. In particular, we provide a way to identify *dominant shared terms*, i.e., terms that are frequently used in different domains, and to measure the variation of meaning of these terms when they are used by different domain experts. We systematically evaluate the method on a set of seven scenarios involving combinations of five different domains. Although in some scenarios the methods appear effective in ranking terms by their cross-domain ambiguity, we observed several cases of inaccuracy, as well as discrepancy between evaluations performed by different human assessors. These cases are due to several factors, including: the presence of terms representing high-level abstract concepts, e.g., *consequence*, which do not vary their meaning regardless of their linguistic context; the presence of in-domain ambiguous terms, i.e., terms that are used with different meanings also within a domain; problems related to the process adopted for evaluating the method, as, e.g., the interpretation of instructions by the human assessors. Our planned future work include:

1. extension of the text pre-processing and the language modelling components to the identification of multi-word expressions, e.g., compound names and collocations;
2. systematic experiments oriented to parameter tuning, notably h, k and ρ in our algorithms. These systematic experiments, especially on variations of k and ρ, will be mainly oriented to understand how the lists of dominant shared terms may vary[13];
3. definition of strategies to identify in-domain ambiguities: to address this goal, we plan to employ more specialised domain-specific corpora extracted from scientific articles available from arXiv[14]. These corpora will be used to train domain-specific word embeddings, possibly focused on more restricted knowledge areas. This solution is also expected to address problems related to the encyclopedic language used by Wikipedia.

References

Agirre E, Edmonds P (2007) Word sense disambiguation: Algorithms and applications, vol 33. Springer Science & Business Media, DOI http://dx.doi.org/10.1007/978-1-4020-4809-8

Agresti A (2010) Analysis of ordinal categorical data, vol 656. John Wiley & Sons

[13] Preliminary experiments were performed in this direction, to obtain the currently adopted parameters' values. However, we reckon that a systematic evaluation campaign is required, as there may be an optimal parameters' selection for each domain group.

[14] `https://arxiv.org`

Ambriola V, Gervasi V (2006) On the systematic analysis of natural language requirements with Circe. ASE 13(1), DOI https://doi.org/10.1007/s10515-006-5468-2

Arora C, Sabetzadeh M, Briand L, Zimmer F (2015) Automated checking of conformance to requirements templates using natural language processing. TSE 41(10):944–968, DOI https://doi.org/10.1109/TSE.2015.2428709

Avramidis E (2013) Rankeval: Open tool for evaluation of machine-learned ranking. The Prague Bulletin of Mathematical Linguistics 100:63–72, DOI https://doi.org/10.2478/pralin-2013-0012

Berry D, Kamsties E (2004) Ambiguity in requirements specification. In: Perspectives on Software Requirements, Springer US, pp 7–44, DOI https://doi.org/10.1007/978-1-4615-0465-8_2

Berry DM, Kamsties E (2005) The syntactically dangerous all and plural in specifications. IEEE Software 22(1):55–57, DOI https://doi.org/10.1109/MS.2005.22

Berry DM, Kamsties E, Krieger MM (2003) From contract drafting to software specification: Linguistic sources of ambiguity. URL `https://cs.uwaterloo.ca/~dberry/handbook/ambiguityHandbook.pdf`

Camacho-Collados J, Pilehvar T (2018) From word to sense embeddings: A survey on vector representations of meaning. arXiv preprint arXiv:180504032 URL `https://arxiv.org/abs/1805.04032`

Casamayor A, Godoy D, Campo M (2012) Functional grouping of natural language requirements for assistance in architectural software design. KBS 30:78–86, DOI https://doi.org/10.1016/j.knosys.2011.12.009

Chantree F, Nuseibeh B, De Roeck AN, Willis A (2006) Identifying nocuous ambiguities in natural language requirements. In: RE'06, pp 56–65, DOI https://doi.org/10.1109/RE.2006.31

Chen X, Liu Z, Sun M (2014) A unified model for word sense representation and disambiguation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp 1025–1035, DOI http://dx.doi.org/10.3115/v1/D14-1110

Cleland-Huang J (2015) Mining domain knowledge [requirements]. IEEE Software 32(3):16–19, URL `https://doi.org/10.1109/MS.2015.67`

Natt och Dag J, Gervasi V, Brinkkemper S, et al. (2005) A linguistic-engineering approach to large-scale requirements management. IEEE software (1):32–39, DOI https://doi.org/10.1109/MS.2005.1

Dalpiaz F, van der Schalk I, Lucassen G (2018) Pinpointing ambiguity and incompleteness in requirements engineering via information visualization and nlp. In: International Working Conference on Requirements Engineering: Foundation for Software Quality, Springer, pp 119–135, DOI https://doi.org/10.1007/978-3-319-77243-1_8

Dieste O, Juristo N (2011) Systematic review and aggregation of empirical studies on elicitation techniques. IEEE Transactions on Software Engineering 37(2):283–304, DOI https://doi.org/10.1109/TSE.2010.33

Evans MC, Bhatia J, Wadkar S, Breaux TD (2017) An evaluation of constituency-based hyponymy extraction from privacy policies. In: Require-

ments Engineering Conference (RE), 2017 IEEE 25th International, IEEE, pp 312–321, DOI https://doi.org/10.1109/RE.2017.87

Falessi D, Cantone G, Canfora G (2013) Empirical principles and an industrial case study in retrieving equivalent requirements via natural language processing techniques. IEEE Transactions on Software Engineering 39(1):18–44, DOI https://doi.org/10.1109/TSE.2011.122

Femmer H, Kučera J, Vetrò A (2014) On the impact of passive voice requirements on domain modelling. In: ESEM, ACM, p 21, DOI https://doi.org/10.1145/2652524.2652554

Femmer H, Fernández DM, Wagner S, Eder S (2017) Rapid quality assurance with requirements smells. JSS 123:190–213, DOI https://doi.org/10.1016/j.jss.2016.02.047

Fernández DM, Wagner S, Kalinowski M, Felderer M, Mafra P, Vetrò A, Conte T, Christiansson MT, Greer D, Lassenius C, Männistö T, Nayabi M, Oivo M, Penzenstadler B, Pfahl D, Prikladnicki R, Ruhe G, Schekelmann A, Sen S, Spinola R, Tuzcu A, de la Vara JL, Wieringa R (2017) Naming the pain in requirements engineering. Empirical Software Engineering 22(5):2298–2338, DOI https://doi.org/10.1007/s10664-016-9451-7

Ferrari A, Gnesi S (2012) Using collective intelligence to detect pragmatic ambiguities. In: RE'12, IEEE, pp 191–200, DOI https://doi.org/10.1109/RE.2012.6345803

Ferrari A, Spoletini P, Gnesi S (2016) Ambiguity and tacit knowledge in requirements elicitation interviews. REJ 21(3):333–355, DOI https://doi.org/10.1007/s00766-016-0249-3

Ferrari A, DellOrletta F, Esuli A, Gervasi V, Gnesi S (2017a) Natural language requirements processing: a 4D vision. IEEE Software (6):28–35, DOI https://doi.org/10.1109/MS.2017.4121207

Ferrari A, Donati B, Gnesi S (2017b) Detecting domain-specific ambiguities: an NLP approach based on Wikipedia crawling and word embeddings. In: 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW), IEEE, pp 393–399, DOI https://doi.org/10.1109/REW.2017.20

Ferrari A, Esuli A, Gnesi S (2018a) Identification of cross-domain ambiguity with language models. In: Groen EC, Harrison R, Murukannaiah PK, Vogelsang A (eds) 5th International Workshop on Artificial Intelligence for Requirements Engineering, AIRE@RE 2018, Banff, AB, Canada, August 21, 2018, IEEE, pp 31–38, DOI https://doi.org/10.1109/AIRE.2018.00011

Ferrari A, Gori G, Rosadini B, Trotta I, Bacherini S, Fantechi A, Gnesi S (2018b) Detecting requirements defects with NLP patterns: an industrial experience in the railway domain. Empirical Software Engineering pp 1–50, DOI https://doi.org/10.1007/s10664-018-9596-7

Firth JR (1968) Selected papers of JR Firth, 1952-59. Indiana University Press

Flekova L, Gurevych I (2016) Supersense embeddings: A unified model for supersense interpretation, prediction, and utilization. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), vol 1, pp 2029–2041, DOI http://dx.doi.org/10.18653/v1/P16-1191

Gacitua R, Sawyer P, Gervasi V (2010) On the effectiveness of abstraction identification in requirements engineering. In: Proceedings of the 18th IEEE International Requirements Engineering Conference (RE'10), IEEE, pp 5–14, DOI http://doi.ieeecomputersociety.org/10.1109/RE.2010.12

Gleich B, Creighton O, Kof L (2010) Ambiguity detection: Towards a tool explaining ambiguity sources. In: REFSQ'10, Springer, LNCS, vol 6182, pp 218–232, DOI https://doi.org/10.1007/978-3-642-14192-8_20

Guo J, Cheng J, Cleland-Huang J (2017) Semantically enhanced software traceability using deep learning techniques. In: Software Engineering (ICSE), 2017 IEEE/ACM 39th International Conference on, IEEE, pp 3–14, DOI https://doi.org/10.1109/ICSE.2017.9

Harris ZS (1954) Distributional structure. Word 10(2-3):146–162, DOI https://doi.org/10.1080/00437956.1954.11659520

Hoffman RR, Shadbolt NR, Burton AM, Klein G (1995) Eliciting knowledge from experts: A methodological analysis. Organ Behav Hum Dec 62(2):129–158, DOI https://doi.org/10.1006/obhd.1995.1039

Jha N, Mahmoud A (2017) Mining user requirements from application store reviews using frame semantics. In: International Working Conference on Requirements Engineering: Foundation for Software Quality, Springer, pp 273–287, DOI https://doi.org/10.1007/978-3-319-54045-0_20

Kendall MG (1945) The treatment of ties in ranking problems. Biometrika 33(3):239–251, DOI https://doi.org/10.2307/2332303

Kof L (2010) From requirements documents to system models: A tool for interactive semi-automatic translation. In: 2010 18th IEEE International Requirements Engineering Conference, pp 391–392, DOI 10.1109/RE.2010.53

Lami G, Gnesi S, Fabbrini F, Fusani M (2001) The linguistic approach to the natural language requirements quality: Benefit of the use of an automatic tool. In: Proceedings 26th Annual NASA Goddard Software Engineering Workshop(SEW), vol 00, p 97, DOI doi.ieeecomputersociety.org/10.1109/SEW.2001.992662

Landis JR, Koch GG (1977) An application of hierarchical kappa-type statistics in the assessment of majority agreement among multiple observers. Biometrics pp 363–374, DOI https://doi.org/10.2307/2529786

Lee YK, Ng HT (2002) An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In: Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10, Association for Computational Linguistics, pp 41–48, DOI https://doi.org/10.3115/1118693.1118699

Levy O, Goldberg Y (2014) Neural word embedding as implicit matrix factorization. In: Adv Neural Inf Process Syst, pp 2177–2185, URL `http://papers.nips.cc/paper/5477-neural-word-embedding-as-implicit-matrix-factorization.pdf`

Lian X, Rahimi M, Cleland-Huang J, Zhang L, Ferrai R, Smith M (2016) Mining requirements knowledge from collections of domain documents. In:

Requirements Engineering Conference (RE), 2016 IEEE 24th International, IEEE, pp 156–165, DOI https://doi.org/10.1109/RE.2016.50

Maalej W, Nabil H (2015) Bug report, feature request, or simply praise? on automatically classifying app reviews. In: Proceedings of the 23rd IEEE International Requirements Engineering Conference, (RE'15), IEEE, pp 116–125, DOI https://doi.org/10.1109/RE.2015.7320414

Massey AK, Rutledge RL, Anton AI, Swire PP (2014) Identifying and classifying ambiguity for regulatory requirements. In: RE'14, IEEE, pp 83–92, DOI https://doi.org/10.1109/RE.2014.6912250

Mavin A, Wilkinson P, Harwood A, Novak M (2009) Easy approach to requirements syntax (EARS). In: RE'09, IEEE, pp 317–322, DOI https://doi.org/10.1109/RE.2009.9

Mich L (1996) NL-OOPS: from natural language to object oriented requirements using the natural language processing system LOLITA. NLE 2(2):161–187, DOI https://doi.org/10.1017/S1351324996001337

Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Adv Neural Inf Process Syst, pp 3111–3119, URL `https://arxiv.org/abs/1310.4546`

Miller G (1998) WordNet: An electronic lexical database. MIT press

Navigli R (2009) Word sense disambiguation: A survey. ACM computing surveys (CSUR) 41(2):10, DOI https://doi.org/10.1145/1459352.1459355

Pedersen T (2000) A simple approach to building ensembles of naive bayesian classifiers for word sense disambiguation. In: Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference, Association for Computational Linguistics, pp 63–69, URL `http://www.aclweb.org/anthology/A00-2009`

Pedersen T (2007) Unsupervised corpus-based methods for wsd. In: Word sense disambiguation, Springer, pp 133–166, DOI https://doi.org/10.1007/978-1-4020-4809-8_6

Pohl K, Rupp C (2011) Requirements engineering fundamentals. Rocky Nook, Inc.

Quirchmayr T, Paech B, Kohl R, Karey H (2017) Semi-automatic software feature-relevant information extraction from natural language user manuals. In: Proceedings of the 23rd International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'17), Springer, pp 255–272, DOI https://doi.org/10.1007/978-3-319-54045-0_19

Raganato A, Bovi CD, Navigli R (2017) Neural sequence learning models for word sense disambiguation. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp 1156–1167, DOI http://dx.doi.org/10.18653/v1/D17-1120

Robeer M, Lucassen G, van der Werf JME, Dalpiaz F, Brinkkemper S (2016) Automated extraction of conceptual models from user stories via nlp. In: Proceedings of the 24th IEEE International Requirements Engineering Conference (RE'16), IEEE, pp 196–205, DOI https://doi.org/10.1109/RE.2016.40

Rodriguez DV, Carver DL, Mahmoud A (2018) An efficient wikipedia-based approach for better understanding of natural language text related to user requirements. In: 2018 IEEE Aerospace Conference, IEEE, pp 1–16, DOI https://doi.org/10.1109/AERO.2018.8396645

Rosadini B, Ferrari A, Gori G, Fantechi A, Gnesi S, Trotta I, Bacherini S (2017) Using NLP to detect requirements defects: An industrial experience in the railway domain. In: REFSQ, Springer, pp 344–360, DOI https://doi.org/10.1007/978-3-319-54045-0_24

Sleimi A, Sannier N, Sabetzadeh M, Briand L, Dann J (2018) Automated extraction of semantic legal metadata using natural language processing. In: 2018 IEEE 26th International Requirements Engineering Conference (RE), IEEE, pp 124–135, DOI https://doi.org/10.1109/RE.2018.00022

Sultanov H, Hayes JH (2013) Application of reinforcement learning to requirements engineering: requirements tracing. In: Proceedings of the 21st IEEE International Requirements Engineering Conference (RE'13), IEEE, pp 52–61, DOI https://doi.org/10.1109/RE.2013.6636705

Taghipour K, Ng HT (2015) Semi-supervised word sense disambiguation using word embeddings in general and specific domains. In: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp 314–323, DOI http://dx.doi.org/10.3115/v1/N15-1035

Tjong S, Berry D (2013) The design of SREE a prototype potential ambiguity finder for requirements specifications and lessons learned. In: REFSQ'13, LNCS, vol 7830, pp 80–95, DOI https://doi.org/10.1007/978-3-642-37422-7_6

Yang H, De Roeck AN, Gervasi V, Willis A, Nuseibeh B (2011) Analysing anaphoric ambiguity in natural language requirements. Requir Eng 16(3):163–189, DOI https://doi.org/10.1007/s00766-011-0119-y

Yuan D, Richardson J, Doherty R, Evans C, Altendorf E (2016) Semi-supervised word sense disambiguation with neural models. arXiv preprint arXiv:160307012 URL `https://arxiv.org/abs/1603.07012`

Yue T, Briand LC, Labiche Y (2015) aToucan: an automated framework to derive UML analysis models from use case models. ACM Transactions on Software Engineering and Methodology (TOSEM) 24(3):13, DOI https://doi.org/10.1145/2699697

Zowghi D, Coulin C (2005) Requirements elicitation: A survey of techniques, approaches, and tools. In: Engineering and managing software requirements, Springer, pp 19–46, DOI https://doi.org/10.1007/3-540-28244-0_2