Identification of Cross-domain Ambiguity with Language Models

Alessio Ferrari*, Andrea Esuli* and Stefania Gnesi*
* CNR-ISTI, Pisa, Italy, Email: alessio.ferrari@isti.cnr.it, andrea.esuli@isti.cnr.it, stefania.gnesi@isti.cnr.it

***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. 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 word, thus estimating its potential ambiguity across the domains of interest. The proposed strategy can be useful to prepare lists of dangerous terms to take into account during requirements elicitation meetings, such as workshops, or focus groups, when these involve stakeholders from distant domains.**

I. 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 [1]–[3]. To this end, requirements elicitation meetings in the form of focus groups, workshops or interviews are organised with the stakeholders [4], [5]. During these meetings, domain experts may use specialised jargons, and ambiguity may occur between requirements 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 [6].

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 [7], [8], adjectives and adverbs [8]–[11], and passive voice [11]–[13]. Other works focus on syntactic ambiguities, which are caused by ambiguous sentence structures [14]. In particular, tools have been provided for the detection of coordination ambiguities, i.e., ambiguities triggered by "and" or "or" conjunctions [15] and anaphoric ones, i.e., ambiguities triggered by pronouns [16]. 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 [17].

In a recent contribution to the AIRE'17 workshop, we proposed one of the first approaches to address domain-dependent ambiguities, and we defined a technique to estimate the ambiguity potential of typical computer science words (e.g., *system*, *database*, *interface*) when they are used in different domains [18]. The proposed approach was focused on the single domain of computer science, and did not consider the ambiguities that could be triggered by words used by experts from other domains. This paper extends our initial idea with a more general approach to compare terminology from an arbitrary number of domains, identify potential cross-domain ambiguities, and overcome some technical limitations of our previous work.

The proposed approach is based on *word embeddings* [19], 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 both computer science and medicine such as *cell* or *function*. 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. An ambiguity ranking is finally produced based on the ambiguity potential of the terms.

To showcase the proposed method, we apply it to a set of hypothetical, yet realistic, pilot scenarios concerning the development of different products, and involving stakeholders from multiple domains. We provide a critical analysis of the results, and we outline future work based on our observations.

The remainder of the paper is structured as follows. In Sect. II we provide some background on ambiguity and word embeddings, and we summarise our previous work[1]. In Sect. III we outline the proposed approach. In Sect. IV we present some results on our pilot scenarios, and Sect. V provides final remarks.

[1]Sect. II-A and part of II-B report content from our previous paper [18], since the background was applicable also to the current one. We considered it useful to include the content for the sake of clarity.

II. Background

A. Ambiguity 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 [20]: *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 [21]: (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.* [20] considers also *language errors* as a separate class of ambiguity. In addition, two other phenomena closely related to ambiguity are discussed in the literature [22], 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.* [20] and Ferrari *et al.* [17].

As highlighted, among others, by Massey *et al.* [23], 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?.

B. 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 [24], 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* [25].

Among the various word embeddings techniques, one of the most widely known was proposed by Mikolov *et al.* [19]. 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 Yoav [26]. 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 $\vec{w}$, where $\vec{w} \in \mathbb{R}^d$, 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 $\vec{h}_w$ 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 $\vec{c}_w$ on the vocabulary V, with ones only in the positions associated to words in c_w. The vector $\vec{h}_w$ 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 $\vec{w} = W^T v_w$ is then multiplied to a second matrix W' of size $d \cdot |V|$, producing $\vec{u} = W'^T \vec{w}$. The softmax function is applied to $\vec{u}$ producing the final prediction of the probabilities of words to appear in the context of the input word $\tilde{c}_y = \mathrm{softmax}(\vec{u})$, which is then compared to $\vec{c}_w$ 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 word embeddings define the *language model* of 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.

C. Detecting Ambiguity of Computer Science Words

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 [18]. 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.

The current paper extends our initial idea 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*. These are the words that are *common* between the different domains of the set, and that are *frequent* in all 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 approach proposed in the current paper also overcomes the technical limitations of our previous work.

III. 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 Intersection**). 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 language model, together with their similarity value. Hence, given a dominant shared term, we compare the lists of its most similar words in 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.

A. 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, we suggest the reader to access the portal for CS[2]. We automatically crawl the different pages of the domains, and we download them to create the corpora. Then, these are pre-processed, by means of stop-word removal and lemmatization. For more details, the reader should refer to Ferrari *et al.* [18].

B. Language Models Generation

In this step, domain-specific language models M_i are learned by means of the `word2vec` algorithm [19], 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 $d = 50$, $L = 10$, $m = 5$. These values have been selected based on preliminary experiments on the data.

C. Cross-domain Intersection

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 below.

INTERSECT($\mathcal{C}, k$)
  $T_{\mathcal{D}} \leftarrow \emptyset$
  $R_{\mathcal{D}}[1 \dots |\mathcal{C}|] \leftarrow [\emptyset \dots \emptyset]$
  $j \leftarrow k$

  for $C_i \in \mathcal{C}$
    do $R_{\mathcal{D}}[i] \leftarrow$ FREQ-RANK(C_i)

  repeat
    do $T_{\mathcal{D}} \leftarrow R_{\mathcal{D}}[1][1 \dots j] \cap \dots \cap R_{\mathcal{D}}[|\mathcal{C}|][1 \dots j]$
      $j \leftarrow j + 1$

  until $|T_{\mathcal{D}}| < k$

  return $T_{\mathcal{D}}$

[2]https://en.wikipedia.org/wiki/Category:Computer_science
[3]In the pseudo-code, array indexes start from 1.

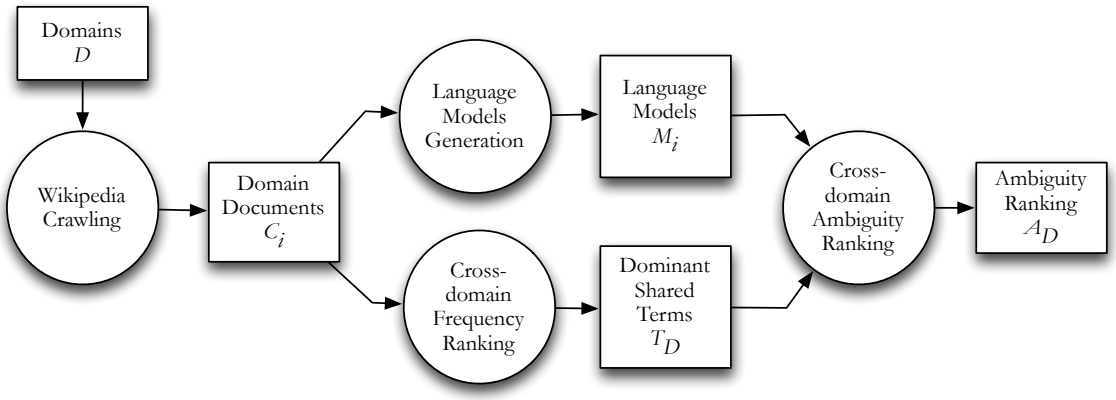

Fig. 1: Approach to Measure Domain-specific Ambiguity

The algorithm takes as input a parameter k, which is the preferred size of $T_{\mathcal{D}}$, and the set of domain specific corpora $\mathcal{C} = \{C_i : i = 1 \dots n\}$. It first ranks the terms of the vocabulary of C_i based on their frequency in C_i (procedure FREQ-RANK(C_i)). This produces a ranked lists of terms for each C_i. The ranked lists are stored in the array $R_{\mathcal{D}}$. Finally, the intersection between the top terms of the ranked lists is incrementally computed, until the size of the intersection reaches the value k.

D. 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., *blood* appears in the similarity list for *cell* 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 in 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 below.

AMBIGUITY-RANK($T_{\mathcal{D}}, \mathcal{M}, h$)
  $A_{\mathcal{D}}, L_i, S_i, U, R, V, \sigma \leftarrow \emptyset$
  for $t \in T_{\mathcal{D}}$
    do
      for $M_i \in \mathcal{M}$
        do $L_i[t], S_i[t] \leftarrow$ MOST-SIMILAR(t, M_i, h)
      $U[t] \leftarrow L_1[t] \cup \dots \cup L_n[t]$
      for $w \in U[t]$
        do
          $R[t][w] \leftarrow$ BEST-RANK($L_1[t], \dots, L_n[t]$)
          $\sigma[t][w] \leftarrow$ VAR($S_1[t][w], \dots, S_n[t][w]$)
      $V[t] \leftarrow \sum_{w \in U[t]} \frac{\sigma[t][w]}{R[t][w]}$
  $A_{\mathcal{D}} \leftarrow$ SORT($T_{\mathcal{D}}, V$)
  return $A_{\mathcal{D}}$

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

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

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.

IV. PILOT SCENARIOS

To provide an early, exploratory evaluation of the proposed approach, we defined a set of potential product development scenarios involving multiple domain experts, and we computed the ranked lists of ambiguous terms for the scenarios. Based on the ranked lists of ambiguous terms obtained, we inspect the Wikipedia pages of the different domains, and show variations of meaning of the terms.

A. 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 scenarios are briefly described 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) **Medical Software** [CS, MED]: a software that supports the prediction of certain diseases, based on symptoms.
2) **Medical Device** [CS, EEN, MED]: a medical device for monitoring the hearth-rate of the patient, and linked to a mobile app.
3) **Medical Robot** [CS, EEN, MEN, MED]: a computer-controlled robot to perform surgery.
4) **Sport Rehab Machine** [CS, EEN, MEN, MED, SPO]: a high-tech machine for rehab, specifically oriented to athletes.

B. Implementation and Application of the Approach

We implemented the approach in Python, using a set of support libraries, namely the 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 reproduce our experiments, can be downloaded from our GIT repository[8]. The repository includes also the generated language models.

In the application of the approach, we downloaded 10,000 Wikipedia pages from each domain-specific Wikipedia portal involved in the scenarios (CS, EEN, MEN, MED, SPO). From our previous experiments [18], 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. III-B. Given the domain-specific corpora, five different language models are generated. Then, the procedures INTERSECT and AMBIGUITY-RANK are applied for each scenario. For our pilots, we set $k = 100$, i.e., the number of dominant shared terms that we wish to rank, and $h = 100$, i.e., the length of the similarity lists that we compare with the AMBIGUITY-RANK procedure. Future work will be performed to fine-tune the values of the parameters h and k. With these settings, we produce the lists of dominant shared terms, ranked by their ambiguity score in the context of each scenario. To check that terms that are ranked higher have a diverse meanings in the domains involved, we check the lists of most similar words for each domain-specific language model, and show the usage of the terms in the Wikipedia pages from the domain-specific portals.

[5]https://pypi.org/project/wikipedia/
[6]https://radimrehurek.com/gensim/
[7]https://spacy.io
[8]https://github.com/isti-fmt-nemis/Domain-specific-ambiguity

C. Results and Discussion

Table I reports the lists of dominant shared terms for each scenario, ranked by ambiguity score. For the sake of space, we report the top-20 and the bottom-20 in the ranked lists. Higher ranks indicate higher chance of ambiguity. Here, we discuss notable cases, based on the different scenarios, and we suggest future directions of research based on our qualitative findings.

a) High-score Terms: We see that the term *surface* appears among the most ambiguous terms in all the scenarios. If we check the most similar (i.e., related) words from the different language models we have the following results. 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 [17]. 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*.

Also the term *cell* has a high ambiguity score in most of the different scenarios. Similar words in different example domains are as follows. MED: *lymphocyte*, *mitosis*, *leukocyte*; EEN: *photovoltaic*, *battery*, *solar*. This is a typical case of lexical ambiguity, since the term *cell* in MED is the structural

TABLE I: Lists of dominant shared terms ranked by ambiguity score for each pilot scenario.

Medical Software		**Medical Device**		**Medical Robot**		**Sport Rehab Machine**	
Term	**Score**	**Term**	**Score**	**Term**	**Score**	**Term**	**Score**
cell	1.29710216	**surface**	1.811224102	**surface**	1.960089321	**surface**	2.229032903
surface	1.296393706	**cell**	1.566944024	**cell**	1.638133097	material	1.947538217
result	1.127396846	theory	1.42422561	law	1.634110549	result	1.908959291
theory	1.089350154	institute	1.415128062	tool	1.594085253	law	1.775950008
function	1.062322513	result	1.343340092	theory	1.578640437	network	1.762205559
device	1.04819773	study	1.301081354	program	1.57862187	combination	1.756012675
agent	1.047093732	signal	1.277533119	material	1.574310743	city	1.737238638
image	1.005565895	**function**	1.276650055	result	1.554693367	**device**	1.717853927
solution	0.967675839	pattern	1.269213667	frequency	1.542611933	element	1.711385627
condition	0.963166843	law	1.267676033	pattern	1.487170693	program	1.709378323
study	0.925899947	board	1.265214143	basis	1.486336771	ability	1.700525242
program	0.913275003	image	1.261162643	environment	1.480037054	film	1.68654784
service	0.909640412	network	1.231872849	signal	1.455549241	study	1.682593029
environment	0.903271417	series	1.226724685	study	1.443556296	practice	1.681793735
activity	0.884836858	rate	1.223808737	image	1.439766954	feature	1.664517617
point	0.884598807	sample	1.21887005	**function**	1.425638406	rate	1.649034779
level	0.87425271	feature	1.206133412	region	1.408171485	service	1.628529767
structure	0.868619609	environment	1.202292721	network	1.393386251	region	1.624247019
university	0.868045926	region	1.177954088	**device**	1.389364284	design	1.617064055
application	0.865258721	power	1.167153435	service	1.359043977	version	1.614382429
...	...	...	...	...	...	...	...
science	0.470753349	number	0.669080294	type	0.881663378	cost	1.049379155
field	0.46829851	day	0.660457815	work	0.869970083	type	1.046449951
education	0.459179568	work	0.656936226	addition	0.864941633	change	1.040798403
group	0.451327105	size	0.651291086	case	0.860270554	number	1.03533031
control	0.45083519	system	0.643853168	group	0.82603429	group	1.012731714
book	0.448669788	concept	0.633515627	method	0.821052665	support	1.008322771
degree	0.434284797	area	0.631627826	system	0.817994908	test	0.988661463
school	0.419147662	control	0.614920953	change	0.806658237	size	0.984698436
area	0.418915449	state	0.606973644	number	0.796776048	control	0.951273998
product	0.385983269	product	0.602132576	concept	0.777989559	article	0.945138259
report	0.383933961	degree	0.601655458	control	0.766416496	market	0.940699417
article	0.336694386	science	0.596553422	day	0.747203241	day	0.881921481
issue	0.333983174	issue	0.592089083	test	0.740795123	period	0.81013248
history	0.330724481	example	0.58672054	cost	0.719874457	state	0.802503629
student	0.321891079	group	0.507565373	state	0.654249723	variety	0.711783266
award	0.319464908	year	0.427463097	example	0.581528382	example	0.695762664
time	0.308453771	award	0.403603996	year	0.550753799	**term**	0.664942784
term	0.214670093	**term**	0.383709019	time	0.478257067	year	0.651922096
range	0.18709287	time	0.381181582	**term**	0.443819094	time	0.568577055
year	0.187065985	**range**	0.288589908	**range**	0.39932581	**range**	0.530688792

unit of a living thing, while in EEN can be the cell of a solar panel, or a battery cell. This latter is an interesting case of intra-domain ambiguity, in which a term is used with different meanings in the same domain. Other techniques are needed to study these cases. The various uses of the term *cell* can be again identified by looking at the sentences in different Wikipedia pages. In CS, we find: *In computing, an address space defines a range of discrete addresses, each of which may correspond to [...] a memory cell or other logical or physical entity.* In MEN, we have: *[...] the resultant shear force [is] measured using a load cell*. In EEN: *[...] techniques that maximize the capacity of a battery pack with multiple cells in series*. In MED: *[...] it often requires ongoing metabolic activity and division of bacterial cells*.

It is worth noting that the term *cell* does not appear in the list for the Sport Rehab Machine case. Indeed, in this case, the SPO domain is involved, in which the term *cell* is not sufficiently frequent to be considered among the set of dominant shared terms. In the (fictional) scenario, this means that the SPO expert will be less likely to use the term *cell* than the other stakeholders. However, the term may be used by the others, and may be the source of ambiguity. This indicates that, given a scenario, to have a complete list of potentially ambiguous terms, one should consider also pairwise cases. An analysis in this sense is left for future work.

Other interesting terms, which appear highly ranked in most of the scenarios are *function* and *device*. Examples of their usage in the different domains are as follows. MED: *An electronic device called a rheoencephalograph [...] is utilized in brain blood flow biofeedback*; SPO: *Game Bike is the name of an interactive fitness device*. CS: *[...] is a mathematical function that is zero-valued outside of some chosen interval*. EEN: *A transfer function that is closer to Weber's law allows for a larger dynamic range*.

The examples show that the abstract terms *function* and *device* may be regarded as potential sources of generality, since their meaning becomes clear when they are associated to

more concrete specifiers (i.e., *electronic* or *fitness* for device; *mathematical* or *transfer* for function).

b) Low-score Terms: The terms at the bottom of the lists in Table I are those that obtain the lowest ambiguity score. For example, the term *range* appears to be one of the least ambiguous, and its most related words in the different domains are *wide, variety, spectrum, broad*, etc. The set of words is consistent across the domains. Similarly, 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, similarly to what was observed for *cell* in ELE. Again, further solutions are required to account for these cases.

V. 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 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 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. Furthermore, we showcase the method on four example scenarios. Given the promising results, our planned future work include: (1) validation of the approach: given a term, a random set of domain-specific documents including the term will be selected to systematically assess whether the term is used with different meanings; (2) systematic evaluation of pairwise similarity among domains. In the current work, we treated the problem of groups of domains, and in future work we wish to measure the impact of domain subgroups; (3) systematic experiments oriented to parameter tuning, notably h and k in our algorithms; (4) definition of strategies to identify intra-domain ambiguities: to address this goal, we plan to employ more specialised domain-specific corpora extracted from scientific articles available from arXiv[9]. These corpora will be used to train domain-specific word embeddings, possibly focused on more restricted knowledge areas.

Acknowledgment

This work was partially supported by the H2020-S2RJU-OC-2017 Project ASTRail (777561).

References

[1] R. R. Hoffman, N. R. Shadbolt, A. M. Burton, and G. Klein, "Eliciting knowledge from experts: A methodological analysis," *Organ. Behav. Hum. Dec.*, vol. 62, no. 2, pp. 129–158, 1995.

[2] J. Cleland-Huang, "Mining domain knowledge [requirements]," *IEEE Software*, vol. 32, no. 3, pp. 16–19, 2015.

[3] X. Lian, M. Rahimi, J. Cleland-Huang, L. Zhang, R. Ferrai, and M. Smith, "Mining requirements knowledge from collections of domain documents," in *Requirements Engineering Conference (RE), 2016 IEEE 24th International*. IEEE, 2016, pp. 156–165.

[4] D. Zowghi and C. Coulin, "Requirements elicitation: A survey of techniques, approaches, and tools," in *Engineering and managing software requirements*. Springer, 2005, pp. 19–46.

[5] O. Dieste and N. Juristo, "Systematic review and aggregation of empirical studies on elicitation techniques," *IEEE Transactions on Software Engineering*, vol. 37, no. 2, pp. 283–304, 2011.

[6] D. M. Fernandez, S. Wagner, M. Kalinowski, M. Felderer, P. Mafra, A. Vetrò, T. Conte, M.-T. Christiansson, D. Greer, C. Lassenius *et al.*, "Naming the pain in requirements engineering," *ESEJ*, pp. 1–41.

[7] D. M. Berry and E. Kamsties, "The syntactically dangerous all and plural in specifications." *IEEE Software*, vol. 22, no. 1, pp. 55–57, 2005.

[8] S. Tjong and D. Berry, "The design of SREE a prototype potential ambiguity finder for requirements specifications and lessons learned," in *REFSQ'13*, ser. LNCS, 2013, vol. 7830, pp. 80–95.

[9] B. Gleich, O. Creighton, and L. Kof, "Ambiguity detection: Towards a tool explaining ambiguity sources," in *REFSQ'10*, ser. LNCS, vol. 6182. Springer, 2010, pp. 218–232.

[10] B. Rosadini, A. Ferrari, G. Gori, A. Fantechi, S. Gnesi, I. Trotta, and S. Bacherini, "Using NLP to detect requirements defects: An industrial experience in the railway domain," in *REFSQ*. Springer, 2017, pp. 344–360.

[11] H. Femmer, D. M. Fernández, S. Wagner, and S. Eder, "Rapid quality assurance with requirements smells," *JSS*, vol. 123, pp. 190–213, 2017.

[12] H. Femmer, J. Kučera, and A. Vetrò, "On the impact of passive voice requirements on domain modelling," in *ESEM*. ACM, 2014, p. 21.

[13] A. Ferrari, G. Gori, B. Rosadini, I. Trotta, S. Bacherini, A. Fantechi, and S. Gnesi, "Detecting requirements defects with nlp patterns: an industrial experience in the railway domain," *Empirical Software Engineering*, pp. 1–50, 2018.

[14] D. Berry and E. Kamsties, "Ambiguity in requirements specification," in *Perspectives on Software Requirements*. Springer US, 2004, pp. 7–44.

[15] F. Chantree, B. Nuseibeh, A. N. D. Roeck, and A. Willis, "Identifying nocuous ambiguities in natural language requirements," in *RE'06*, 2006, pp. 56–65.

[16] H. Yang, A. N. D. Roeck, V. Gervasi, A. Willis, and B. Nuseibeh, "Analysing anaphoric ambiguity in natural language requirements," *Requir. Eng.*, vol. 16, no. 3, pp. 163–189, 2011.

[17] A. Ferrari, P. Spoletini, and S. Gnesi, "Ambiguity and tacit knowledge in requirements elicitation interviews," *REJ*, vol. 21, no. 3, pp. 333–355, 2016.

[18] A. Ferrari, B. Donati, and S. Gnesi, "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, 2017, pp. 393–399.

[19] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Adv Neural Inf Process Syst*, 2013, pp. 3111–3119.

[20] D. M. Berry, E. Kamsties, and M. M. Krieger, "From contract drafting to software specification: Linguistic sources of ambiguity," 2003. [Online]. Available: https://cs.uwaterloo.ca/~dberry/handbook/ambiguityHandbook.pdf

[21] A. Ferrari and S. Gnesi, "Using collective intelligence to detect pragmatic ambiguities," in *RE'12*. IEEE, 2012, pp. 191–200.

[22] F. Fabbrini, M. Fusani, S. Gnesi, and G. Lami, "An automatic quality evaluation for natural language requirements," in *REFSQ'01*, 2001, pp. 150–164.

[23] A. K. Massey, R. L. Rutledge, A. I. Anton, and P. P. Swire, "Identifying and classifying ambiguity for regulatory requirements," in *RE'14*. IEEE, 2014, pp. 83–92.

[24] Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.

[25] J. R. Firth, *Selected papers of JR Firth, 1952-59*. Indiana University Press, 1968.

[26] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," in *Adv Neural Inf Process Syst*, 2014, pp. 2177–2185.

[9] https://arxiv.org