On the Use of Word Embeddings for Identifying Domain Specific Ambiguities in Requirements

Siba Mishra and Arpit Sharma
EECS Department, Indian Institute of Science Education and Research Bhopal, India
{sibam,arpit}@iiserb.ac.in

***Abstract*—Software requirements are usually written in common natural language. An important quality criterion for each documented requirement is unambiguity. This simply means that all readers of the requirement must arrive at the same understanding of the requirement. Due to differences in the domain expertise of requirements engineer and other stakeholders of the project, it is possible that requirements contain several words that allow alternative interpretations. Our objective is to identify and detect domain specific ambiguous words in natural language text. This paper applies an NLP technique based on word embeddings to detect such ambiguous words. More specifically, we measure the ambiguity potential of most frequently used computer science (CS) words when they are used in other application areas or subdomains of engineering, e.g., aerospace, civil, petroleum, biomedical and environmental etc. Our extensive and detailed experiments with several different subdomains show that word embedding based techniques are very effective in identifying domain specific ambiguities. Our findings also demonstrate that this technique can be applied to documents of varying sizes. Finally, we provide pointers for future research.**

***Index Terms*—Requirements, quality, ambiguity, domain, word embedding, natural language.**

I. Introduction

Requirements are the basis for every project, defining what the stakeholders in a potential new system need from it, and also what the system must do in order to satisfy that need [1], [2]. All subsequent steps in software development are influenced by the requirements. Hence, improving the quality of requirements means improving the quality of the product. Requirements are usually written in natural language. Ambiguity is one of the major cause of poor quality requirements document [2], [3]. Ambiguity means that a single reader can interpret the requirement in more than one way or that multiple readers come to different interpretations. The latter could be present because of situations which involve people with different technical backgrounds and domain expertise. Requirements engineer usually has a background in computer science and may use typical computer science words in the requirements document which may be interpreted differently by other stakeholders of the project. This is because these stakeholders may not necessarily have a background in computer science. For example, the word "Windows" means an operating system when used in the context of computer science. On the other hand, for a building engineer, window means a space in the wall of a building. Similarly, for a computer engineer, the word "platform" denotes a complete software programming development environment, whereas for a chemical or petroleum engineer, it means a raised level surface on which people can stand. These ambiguities can lead to time and cost overrun in a software project. In the worst case scenario, these issues can lead to failure of the whole project. Therefore, it is of utmost importance that these issues are detected and resolved in the early phase of software development.

This paper focuses on detecting domain or context specific ambiguities in natural language text. To this end, we use word embeddings for identifying ambiguous words in a document. Word embedding is capable of capturing the context of a word and compute its semantic similarity relation with other words used in a document. It represents individual words as real-valued vectors in a predefined vector space. Each word is mapped to one vector and the vector values are learned based on the usage of words. Words that are used in similar ways tend to have similar representations. This means the distance between two words that are semantically very similar is going to be smaller. More formally, the cosine of the angle between such vectors should be close to 1.

We prepare a list L of most frequently used words in a computer science corpus created by crawling the computer science category on Wikipedia. Similarly, for each subdomain or application area (subcategory) of engineering domain (category), e.g., petroleum, civil, biomedical, environmental and chemical engineering, we create a corresponding corpus by Wikipedia crawling. Next, we estimate the ambiguity potential of words in list L when used in different subdomains of engineering. In other words, for each subdomain we find out which of the commonly used computer science words have a very different meaning and therefore should be cautiously used by the analyst in the requirements document. This technique can also be used for detecting ambiguous words during the requirements elicitation phase. Note that these subdomains have been selected to represent different application areas of engineering for which a computer based problem solving may be required.

The remainder of the paper is structured as follows: Section II discusses the related work. Section III provides the required background. Section IV explains our approach. We present the results and findings in Section V. Finally, Section VI concludes the paper and provides pointers for future research.

978-1-7281-5165-6/19/$31.00 ©2019 IEEE
DOI 10.1109/REW.2019.00048

II. Related Work

For requirements, research has mainly concentrated on identifying ambiguous words, terms and sentences in natural language text [3]–[7]. Note that all these papers focus on domain independent ambiguous words or constructions. Translation of requirements into formal models to resolve the issue of ambiguity has been investigated in [8]–[10]. In [11], indices of ambiguity have been defined and calculated using the knowledge base of natural language processing system. Natural language understanding methodology has also been used for ambiguity detection in [12], [13]. In [14], [15], machine learning techniques have been employed to identify potentially nocuous ambiguities.

The role of contextual and situational ambiguity in requirements elicitation interviews has been extensively studied in [16], [17]. A graph-based approach to detect pragmatic ambiguities in natural language requirements has been proposed in [18], [19]. Both the papers use a graph-based modeling of the background knowledge of different readers, and employ a shortest-path search algorithm to model the pragmatic interpretation of a requirement. Comparison of different pragmatic interpretations determines if a requirement is ambiguous or not.

Recently, an approach based on word embeddings and Wikipedia crawling has been proposed to detect domain specific ambiguities in natural language text [20], [21]. In [20], authors investigate the ambiguity potential of typical computer science words using word2vec algorithm and perform some preliminary experiments. In [21], authors estimate the variation of meaning of dominant shared terms in different domains by comparing the list of most similar words in each domain specific model. This method was applied to some pilot scenarios which involved different products and stakeholders from different domains. Our paper builds on that investigated in [20] and extends it in the following ways :

- Unlike [20] where a few unrelated domains, e.g., medicine, literature, sports and mechanical were selected for experiments, we focus on identifying the ambiguity potential of typical computer science words when used in several different subdomains or application areas of engineering. We create corpora for 13 subdomains of engineering, e.g., aerospace, chemical, civil, biomedical and environmental. We conduct detailed experiments and systematic evaluation of results to identify ambiguous computer science words for each subdomain.
- Unlike [20], we place these subdomains into three different classes, i.e., large, medium and small based on the number of Wikipedia pages crawled to build the corpus. Several rounds of experiments have been performed to identify the most suitable word2vec parameters for every class.
- Based on our experiments, we identify a similarity threshold to detect ambiguous words.
- For every ambiguous computer science word that is present in an engineering subdomain, we report its most similar words and also provide some example sentences from the corpus highlighting its domain specific interpretation.

Our detailed experimental results not only validate the claims made in [20] but also demonstrate the wider applicability of word embeddings based approach for identifying domain specific ambiguities. Additionally, we show that this approach can be applied to documents of varying sizes. This essentially means that it can be used in both small and large software projects.

III. Preliminaries

Word Embeddings: Word embeddings are a powerful approach for analyzing language and have been widely used in information retrieval and text mining. They provide a dense representation of words in the form of numeric vectors which capture the natural semantic relationship of their meaning. Word embeddings are considered to be an improvement over the traditional bag-of-words model which results in very large and sparse word vectors. Out of various word embedding models, the model "word2vec" developed by the researchers at Google [22] has been used in our work to learn and generate word embeddings from a natural language text corpus. We focus on the model named skip gram negative sampling (SGNS) implementation of word2vec [23]. SGNS predicts a collection of words $w \in V_W$ and their contexts $c \in V_C$, where V_W and V_C are the vocabularies of input words and context words respectively. Context words of a word w_i is a set of words $w_{i-wind}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+wind}$ for some fixed window size $wind$. Let D be a multi-set of all word-context pairs observed in the corpus. Let $\vec{w}, \vec{c} \in \mathbb{R}^d$ be the d-dimensional *word embeddings* of word w and context c. These vectors (both word and context) are created by the word2vec model from a corpus and are analyzed to check the semantic similarity between them. The main objective of negative sampling (NS) is to learn high-quality word vector representations on a corpus. A logistic loss function is used in NS for minimizing the negative log-likelihood of words in the training set. For more details, we refer the interested reader to [22]–[24]. As the input corpus changes, word embeddings are also updated reflecting the semantic similarity between words w.r.t. new corpus. This means that for each subdomain, a corresponding language model is generated which can be used to estimate the ambiguity potential of CS words.

IV. Approach

This section discusses the approach used to measure the ambiguity potential of commonly used computer science words.

a) Corpus Building: With the help of some standard web scraping packages in python[1], we crawl and build the corpora of computer science domain C_{CS} and other subdomains of engineering. Let C_{SD} denotes the corpora of a subdomain SD, e.g., corpora for aerospace engineering is denoted by C_{AE}.

[1]https://selenium-python.readthedocs.io/

TABLE I
DESCRIPTIVE STATISTICS OF THE CORPORA.

Category Name	Pages	Sentences	Words
Computer Science (CS)	9021	2,46,359	18,37,492
Building Engineering (BUE)	9002	3,52,005	25,77,515
Mechanical Engineering (MCEE)	7587	3,31,746	24,78,977
Electronic Engineering (ELCE)	7147	2,47,649	18,78,728
Civil Engineering (CIVE)	7071	2,83,337	21,42,500
Aerospace Engineering (AE)	4661	1,61,867	13,13,054
Chemical Engineering (CHEE)	4442	2,03,637	15,37,857
Environmental Engineering (ENVE)	2626	1,16,685	8,72,305
Marine Engineering (MAEE)	1369	31,712	2,23,956
Industrial Engineering (INEE)	1060	42,751	3,41,308
Military Engineering (MLEE)	932	32,068	2,42,944
Biomedical Engineering (BIEE)	924	52,599	3,87,492
Petroleum Engineering (PTEE)	419	15,148	1,21,614
Ceramic Engineering (CERE)	318	12,465	83,705

C_{CS} has been built by collecting the pages from "Wikipedia CS" (computer science) category[2] which has a tree structure. Here, nodes are categories and leaves are pages. Maximum number of pages that can be downloaded for a category is 10000 and the maximum depth used for subcategory traversal is 2. We use aforementioned method to build the corpus for all the subdomains of engineering. For the sake of completeness, we have crosschecked all the results (data extraction for a specific Wikipedia category) with the help of a widely used Wikipedia category data extraction tool known as PetScan[3]. PetScan (previously CatScan) is an external tool which can be used to find all the pages that belong to a Wikipedia category for some specified criteria.

b) Data Pre-processing: This step is useful for generating efficient word embedding vectors and preventing the vocabulary from becoming unnecessarily large. The textual data (sentences) of each corpus are broken into tokens of words (*tokenization*) followed by the cleaning of all special symbols, alpha-numeric words and punctuation marks (*punctuation removal*). Next, we convert all the words to lowercase (*lowering of words*) followed by the removal of noisy words defined for the English language (*stop word removal*). Finally, we lemmatize all the words. Lemmatization removes the inflectional endings and returns the base or dictionary form of a word, i.e., *lemma*[4]. The output of this step produces a new corpora for CS, i.e., $C^{'}_{CS}$ and each subdomain of engineering, i.e., $C^{'}_{SD}$. The descriptive statistics of the collected data after the corpus building and pre-processing steps are shown in Table I. The corpora of all the subdomains have been classified into three classes : 1) large-sized (4401-9500 pages) 2) medium-sized (900-4400 pages) and 3) small-sized (300-899 pages) based on the total number of pages crawled to build the corpora.

c) Word Frequency Count, Intersection and Insertion: We use Part-Of-Speech Tagger (POS Tagger)[5] to identify all the nouns in the corpora. Based on the frequency count, top 100 nouns of each corpora have been selected. Let T_{CS} and T_{SD} denote the sets of top 100 nouns in CS and a subdomain SD. Next, for each subdomain we compute $TC_{SD} = T_{CS} \cap T_{SD}$. Note that for each subdomain SD, we estimate the ambiguity potential of those CS words which belong to the set TC_{SD}. In order to distinguish the corpora, each *lemma* of $C^{'}_{SD}$ is modified to *_lemma_* producing a new corpora $C^{''}_{SD}$.

d) Learning Word Embeddings: Learning of word embeddings is facilitated by joining the two corpora, i.e, $C^{'}_{CS} \cup C^{''}_{SD}$ which is used as input to the word2vec model. We have used three sets of parameters for our experiments based on the size of corpora. We set the dimension ($d = 200$), the window size ($wind = 20$) and the minimum count ($c = 5$) for all the experiments. Additionally, we set the sample[6] ($s = 0.00001$) and the workers[7] ($w = 10$) for all the experiments. Note that other parameters like number of iterations (i) and negative[8] (n) have been fine-tuned according to the size of corpora. In this regard, $n = 25$ for large-sized subdomains and $n = 10$ for medium and small-sized subdomains. Similarly, $i = 3$ for large and medium-sized subdomains and $i = 5$ for small-sized subdomains.

e) Word Similarity Computation: The word2vec model uses the *cosine similarity* to compute the semantic relationship of two different words in vector space. Let us assume two word embedding vectors $\overrightarrow{\boldsymbol{w}'}$ and $\overrightarrow{\boldsymbol{w}''}$, where $\overrightarrow{\boldsymbol{w}'}$ is a word vector generated for CS domain and $\overrightarrow{\boldsymbol{w}''}$ is a word vector for subdomain SD. The cosine angle between these two word embedding vectors is calculated using Equation (1).

$$cos(\overrightarrow{\boldsymbol{w}'}, \overrightarrow{\boldsymbol{w}''}) = \frac{\overrightarrow{\boldsymbol{w}'} \bullet \overrightarrow{\boldsymbol{w}''}}{|\overrightarrow{\boldsymbol{w}'}||\overrightarrow{\boldsymbol{w}''}|} \tag{1}$$

The range of similarity score is between 0 to 1. The scores closer to 1 means that the words are semantically more similar and used in almost the same context. On the other hand, scores closer to 0 means that the words are less related to each other.

V. RESULTS AND FINDINGS

In this section, we present the results of our detailed experiments. We first discuss the results for large-sized subdomains (Section V-A), followed by medium-sized subdomains (Section V-B) and then small-sized subdomains (Section V-C). The approach highlighted in Section IV has been implemented using Python 3.7 along with some supported packages and libraries. Our experiments have been executed on Windows 10 machine with Intel Core i5-7500 CPU, 4GB DDR3 primary memory and a processor frequency of 3.40 GHz. For implementing the word2vec model, we have used the *gensim*[9]

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

[3]https://petscan.wmflabs.org/

[4]https://www.nltk.org/_modules/nltk/stem/wordnet.html

[5]https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

[6]The threshold for configuring which higher-frequency words are randomly down-sampled.

[7]These are the total number of threads needed to train the model.

[8]It indicates the total number of "noisy words" to be drawn. In our experiments, we have used a negative sampling to sample the training data.

[9]https://radimrehurek.com/gensim/

library. For NLP related tasks, the *nltk*[10] package has been used. The parameters of the word2vec model have been adjusted according to the size of the class. For each subdomain, we only report those CS words which are ambiguous. For each subdomain, the similarity threshold used to label a CS word as ambiguous is 0.7. This value has been selected based on our experiments on the corpora.

A. Large-sized Subdomains

1) CS versus BUE Subdomain: The most frequently used CS words which have a different meaning or interpretation in the subdomain of BUE are shown in Table II. Example sentences from the corpora showing the variation of meaning of CS words window and source are given below :

A1. window : An active **window** is the currently focused **window** in the current **window** manager or explorer.
A2. _window_ : An airflow **_window_** is composed of at least two panes of glass and a cavity between them that allows the flow of ventilation air.
B1. source : Auto Keras is an open-**source** python package for neural architecture search.
B2. _source_ : An absorption refrigerator is a refrigerator that uses a heat **_source_**.

TABLE II
SIMILARITY SCORES AND MOST SIMILAR WORDS FOR CS AND BUE.

Words	Similarity Score	Most Similar Words (CS)	Most Similar Words (BUE)
window	0.53485	drive, file, folder, linux, driver	curtain, pit, clothes, fireplace, tile
model	0.38218	method, data, analysis, simulation, prediction	mechanic, law, matter, thermodynamics
source	0.37184	tool, application, specification, library	cooling, earth, noise, generator
device	0.30100	hardware, access, monitor, address	pump, chiller, motor, conditioner
application	0.26846	development, database, software, tool, architecture, source	apparatus, duct, leakage, refrigerant, combustion
control	0.25040	code, source, database, interface, storage, requirement	ventilation, rating, consumption, supply, electricity
design	0.24192	program, network, processing, architecture, knowledge, analysis	edition, issue, communication, access, activity
system	0.12061	network, design, development, software, database, program	power, area, control, ventilation, consumption

2) CS versus MCEE Subdomain: The most frequently used CS words which have a different meaning in the subdomain of MCEE are shown in Table III. Example sentences from the corpora showing the variation of meaning of CS words machine and device are given below :

A1. machine : A finite state **machine** just looks at the input signal and the current state: it has no stack to work with.
A2. _machine_ : An air classifier is an industrial **_machine_** which separates materials by a combination of size, shape, and density.
B1. device : A simple 2D mouse may be considered a navigation **device**.
B2. _device_ : A thermal cutout safety **_device_** is required to prevent overheating of the heating element.

[10]https://www.nltk.org/

TABLE III
SIMILARITY SCORES AND MOST SIMILAR WORDS FOR CS AND MCEE.

Words	Similarity Score	Most Similar Words (CS)	Most Similar Words (MCEE)
device	0.41239	monitor, address, access, hardware	airbag, electricity, fabrication, petroleum, diagnostics
machine	0.36751	data, semantics, pattern, class, complexity, search, analysis	construction, task, manufacturing, category, hazard, diagnostics
design	0.27568	knowledge, network, analysis, program, processing, architecture	standardization, construction, standard, guideline, fabrication, equipment

3) CS versus ELCE Subdomain: The most frequently used CS words which have a different meaning in the subdomain of ELCE are shown in Table IV. Example sentences from the corpora showing the variation of meaning of CS word logic are given below :

A1. Logic : Various types of temporal **logic** can be used to help reason about concurrent systems.
A2. _Logic_ : The other pin is configured as an output and set to the low **_logic_** level.

TABLE IV
SIMILARITY SCORES AND MOST SIMILAR WORDS FOR CS AND ELCE.

Words	Similarity Score	Most Similar Words (CS)	Most Similar Words (ELCE)
logic	0.63814	computation, semantics, turing, geometry, calculus	timing, map, cmos, consumption, parallel, redundancy

4) CS versus CIVE Subdomain: The most frequently used CS words which have a different meaning in the subdomain of CIVE are shown in Table V. Example sentences from the corpora showing the variation of meaning of CS words state and process are given below :

A1. state : The **state** at which the automaton stops is called the final **state**.
A2. _state_ : In 1872, Alexey Von Schmidt undertook the survey of the **_state_** line.
B1. process : In computing, a **process** is an instance of a computer program that is being executed.
B2. _process_ : Infiltration is the **_process_** by which water enters the soil.

TABLE V
SIMILARITY SCORES AND MOST SIMILAR WORDS FOR CS AND CIVE.

Words	Similarity Score	Most Similar Words (CS)	Most Similar Words (CIVE)
state	0.49546	class, algorithm, automaton, model, domain	land, link, highway, survey, government
source	0.47845	library, application, specification, tool	groundwater, recovery, growth, consumption, cycle, storage
product	0.32516	source, mode, cache, code, requirement	chemical, reduction, fabrication, equipment, inspection
process	0.27102	requirement, layer, specification, code, command, memory, storage	digestion, fuel, oil, pollutant, tank, chemical
control	0.26527	code, source, database, interface, storage, requirement	power, quality, safety, impact, hazard

5) CS versus AE Subdomain: The most frequently used CS words which have a different meaning in the subdomain of AE are shown in Table VI. Example sentences from the corpora showing the variation of meaning of CS words system and space are given below :

A1. system : Input devices are instruments used to manipulate objects, and send control instructions to the computer **system**.
A2. _system_ : "9K121" is the GRAU designation for the missile **_system_**.
B1. space : All computation takes both time and **space** (in memory).
B2. _space_ : APAS-89 was envisioned to be the docking system for Buran with the Mir **_space_** station.

TABLE VI
SIMILARITY SCORES AND MOST SIMILAR WORDS FOR CS AND AE.

Words	Similarity Score	Most Similar Words (CS)	Most Similar Words (AE)
space	0.59611	domain, set, regression, solution, element, number, property	mission, launch, spacecraft, shuttle, traffic, safety, satellite
program	0.53640	design, network, knowledge, performance, language	shuttle, station, management, nasa, space, safety
control	0.48789	code, source, database, interface, storage, requirement	power, vehicle, stage, payload, ground, radar, rocket, force
design	0.45121	program, language, machine, mining, processing, programming	cockpit, cabin, motor, airplane, airspace, avionics
service	0.25848	security, access, tool, enterprise, platform, support, user	aircraft, flight, air, mobile-satellite, passenger
data	0.25204	process, analysis, application, code, software	fuel, altitude, level, tank, area, range
system	0.16622	software, data, process, application, program	rocket, radio, vehicle, navigation, radar, power

6) CS versus CHEE Subdomain: The most frequently used CS words which have a different meaning in the subdomain of CHEE are shown in Table VII. Example sentences from the corpora showing the variation of meaning of CS words product and environment are given below :
A1. product : **Product** engineering software is used to help develop large machines and other application software.
A2. _product_ : The **_product_** of the condensation of two molecules of acetic acid is acetic anhydride.
B1. environment : EarSketch is a free educational programming **environment**.
B2. _environment_ : Acetogens have diverse metabolic roles, which help them thrive in different **_environment_**.

TABLE VII
SIMILARITY SCORES AND MOST SIMILAR WORDS FOR CS AND CHEE.

Words	Similarity Score	Most Similar Words (CS)	Most Similar Words (CHEE)
product	0.57344	source, code, cache, mode, requirement	steam, soil, ammonia, combustion, methane, compound
process	0.52690	specification, code, command, layer, requirement, memory, storage	hydrogen, carbon, combustion, water, emission, oxygen
environment	0.50088	driver, share, encryption, resource, database	biodiesel, coal, pollution, impact, waste, treatment
application	0.48448	interface, source, software, access, protocol, package	engine, meter, oxygen, capture, generation

B. Medium-sized Subdomains

1) CS versus ENVE Subdomain: The most frequently used CS words which have a different meaning in the subdomain of ENVE are shown in Table VIII. Example sentences from the corpora showing the variation of meaning of CS words source and tree are given below :
A1. source : Auto Keras is an open-**source** python package for neural architecture search.
A2. _source_ : Cultivated plants are a major **_source_** of adventive populations.
B1. tree : A left-leaning red–black (LLRB) **tree** is a type of self-balancing binary search **tree**.
B2. _tree_ : Van Mahotsav is an annual pan-Indian **_tree_** planting festival.

TABLE VIII
SIMILARITY SCORES AND MOST SIMILAR WORDS FOR CS AND ENVE.

Words	Similarity Score	Most Similar Words (CS)	Most Similar Words (ENVE)
product	0.22886	processing, code, integration, configuration, process, source	chemical, heat, reduction, material, vehicle
tree	0.21638	heap, queue, insertion, sort, hash, merge, algorithm	hydrogen, acid, reaction, biogas, reserve
process	0.17855	database, code, user, analysis, source	methane, soil, biomass, oil, carbon, fuel
source	0.15498	memory, code, file, process, image	production, waste, greenhouse, power, conservation, treatment
state	0.10423	computation, machine, simulation, processing, analysis, complexity	climate, study, environment, waste, production

2) CS versus MAEE Subdomain: The most frequently used CS words which have a different meaning in the subdomain of MAEE are shown in Table IX. Example sentences from the corpora showing the variation of meaning of CS words structure and operation are given below :
A1. structure : The code must also have a clear and clean **structure** so that a human reader can easily understand the algorithm used.
A2. _structure_ : A ballast tank is a compartment within a boat, ship or other floating **_structure_** that holds water.
B1. operation : In abstract algebra, a Robbins algebra is an algebra containing a single binary **operation**.
B2. _operation_ : The first Voith Schneider Propellers were put into **_operation_** in the narrow canals of Venice, Italy.

TABLE IX
SIMILARITY SCORES AND MOST SIMILAR WORDS FOR CS AND MAEE.

Words	Similarity Score	Most Similar Words (CS)	Most Similar Words (MAEE)
structure	0.56361	class, object, method, recursion, regression, procedure	tank, port, dock, yacht, plant, coast
operation	0.51151	class, procedure, optimization, method, structure	war, submarine, sea, propellor, cargo, safety

3) CS versus INEE Subdomain: The most frequently used CS words which have a different meaning in the subdomain of INEE are shown in Table X. Example sentences from the corpora showing the variation of meaning of CS words code and element are given below :
A1. code : Lines of **code** (LOC) was another popular measure of the size of a program.
A2. _code_ : In the USA EDRs must meet federal standards, as described within the U.S. **_code_** of federal regulations.
B1. element : The processor may scale the index register to allow for the size of each array **element**.
B2. _element_ : A signature **_element_** that served to distinguish BMW vehicles on the road, halo headlights were soon borrowed by other automakers.

TABLE X
SIMILARITY SCORES AND MOST SIMILAR WORDS FOR CS AND INEE.

Words	Similarity Score	Most Similar Words (CS)	Most Similar Words (INEE)
tool	0.61881	environment, user, cloud, protocol, resource, database	industry, safety, manufacturing, motor, project, organization
code	0.54016	source, memory, specification, product, format	assembly, policy, collection, economy, market
element	0.35913	input, space, theorem, integer, sequence, string	enterprise, industry, assembly, economics, motor, component

4) CS versus MLEE Subdomain: The most frequently used CS words which have a different meaning in the subdomain of MLEE are shown in Table XI. Example sentences from the corpora showing the variation of meaning of CS words action and operation are given below :

A1. action : Artificial general intelligence (AGI) describes research that aims to create machines capable of general intelligent **action**.
A2. _action_ : George I. Bernay, the first among the unit to be killed in **_action_** (7 Dec 1944).
B1. operation : Write-through **operation** is common when operating over unreliable networks.
B2. _operation_ : The military forces did not fire a shot during the **_operation_**.

TABLE XI
SIMILARITY SCORES AND MOST SIMILAR WORDS FOR CS AND MLEE.

Words	Similarity Score	Most Similar Words (CS)	Most Similar Words (MLEE)
machine	0.65724	process, analysis, code, computation, data	defense, casualty, explosive, explosion, ammunition
action	0.42494	loop, class, boolean, recursion, trace	order, cluster, brigade, training, bomb
operation	0.37621	object, block, integration, procedure, query	combat, hill, infantry, battle, attack
state	0.35903	model, field, number, position, solution	operation, area, center, royal, group

5) CS versus BIEE Subdomain: The most frequently used CS words which have a different meaning in the subdomain of BIEE are shown in Table XII. Example sentences from the corpora showing the variation of meaning of CS words machine, cell and function are given below :

A1. machine : User-accessible registers can be read or written by **machine** instructions.
A2. _machine_ : One of the most successful approaches is an external device that acts similarly to a dialysis **_machine_**.
B1. cell : Apple provided a biogas-powered fuel **cell** and built rooftop solar photovoltaic systems.
B2. _cell_ : Clots formed by red blood **_cell_** and platelet damage can block up blood vessels and lead to very serious consequences.
C1. function : The **function** point metric was introduced to calculate the number of user input and output transactions.
C2. _function_ : Restoration for the volitional motor **_function_** via an artificial neural connection.

TABLE XII
SIMILARITY SCORES AND MOST SIMILAR WORDS FOR CS AND BIEE.

Words	Similarity Score	Most Similar Words (CS)	Most Similar Words (BIEE)
machine	0.50764	process, code, event, computation, source, analysis	stimulation, blood, movement, heart, material
system	0.43792	network, design, software, development, database, program	neuroscience, time, neuron, activity, breast, cell, brain, rate
action	0.38285	operation, decision, configuration, object, pattern, channel	neuron, anatomy, shape, image, flow, membrane
state	0.28745	class, algorithm, automaton, domain, model	activity, breast, rate, neuron, image
cell	0.16522	value, rank, error, size, input	chemical, brain, stem, neuroscience, image
function	0.10232	case, operator, parameter, array, variable, integer, input, define	anatomy, momentum, equation, condition, cell, coordinate

C. Small-sized Subdomains

1) CS versus PTEE Subdomain: The most frequently used CS words which have a different meaning in the subdomain of PTEE are shown in Table XIII. Example sentences from the corpora showing the variation of meaning of CS words platform and tool are given below :

A1. platform : HoneyC is a **platform** independent open source framework written in Ruby.
A2. _platform_ : The first tower emerged in the early 1980s with the installation of Exxon's Lena oil **_platform_**.
B1. tool : Behavior Markup Language (BML) is a **tool** for describing autonomous actor behavior in simulations and computer games.
B2. _tool_ : The caliper **_tool_** measures the variation in borehole diameter.

TABLE XIII
SIMILARITY SCORES AND MOST SIMILAR WORDS FOR CS AND PTEE.

Words	Similarity Score	Most Similar Words (CS)	Most Similar Words (PTEE)
platform	0.60037	editor, email, desktop, apple, interface, sun, gui, firewall	equipment, sea, site, lift, construction, level
tool	0.55951	application, database, protocol, source, web, cloud, document, library	injection, hole, drill, perforation, valve, pump
program	0.49517	design, network, knowledge, language, performance	institute, business, education, center, science, management, society
source	0.43324	library, specification, application, tool	well, jet, heat, yield, hydrogenation, presence
data	0.38112	process, code, analysis, application, software	reservoir, stress, structure, safety, detector, injection
structure	0.35696	class, object, method, optimization	site, rig, sea, platform, reservoir, equipment
operation	0.31831	object, procedure, block, integration, query	surface, sea, area, chemical, valve
state	0.24601	algorithm, class, automaton, model, domain	acetylene, bond, petroleum, refinery, company, carbide

2) CS versus CERE Subdomain: The most frequently used CS words which have a different meaning in the subdomain of CERE are shown in Table XIV. Example sentences from the corpora showing the variation of meaning of CS words application and property are given below :

A1. application : BioUML Workbench is a Java **application** that can work standalone or as "thick client" for the BioUML server edition.
A2. _application_ : The mold is obtained by **_application_** of a bed of clay or plaster on the prototype.

B1. property : One possible route to separating two complexity classes is to find some closure **property** possessed by one and not by the other.

B2. _property_ : The special materials and processes used in powder metallurgy can pose hazards to life and **_property_**.

TABLE XIV
SIMILARITY SCORES AND MOST SIMILAR WORDS FOR CS AND CERE.

Words	Similarity Score	Most Similar Words (CS)	Most Similar Words (CERE)
system	0.48206	task, analysis, specification, language, product	acid, synthesis, ion, reaction, flux, silica, gas
application	0.47303	tool, user, suite, platform, microsoft	water, cement, bone, steel, insulator, chemical, powder
process	0.45779	requirement, task, specification, memory, product, commit	phase, chemical, surface, cement, heat, glaze, range, mineral
structure	0.39269	range, optimization, decision, markov	surface, magnesium, iron, cement, water, crack, mixture
property	0.27801	boundary, scale, boolean, loop	water, silicon, object, carbide, surface, industry
state	0.26670	logic, model, computation, automaton, computer, complexity	construction, mineral, industry, london, manufacture

In this paper we have only focused on detecting domain specific ambiguities but this approach can also be used to identify commonly used CS words or terms that have the same meaning across several different subdomains.

VI. CONCLUSIONS AND FUTURE WORK

This paper demonstrated the applicability of word2vec algorithm for detection of domain specific ambiguity in natural language requirements. We prepared a list of frequently used computer science words and estimated the ambiguity potential of these words when used in different application areas or subdomains of engineering. For every ambiguous word in a subdomain, we reported its most similar words and presented some example sentences to highlight its domain specific interpretation. This research work can be extended in several interesting directions, e.g., one can investigate the feasibility and applicability of this technique to identify domain specific ambiguities in large scale requirements specifications. Similarly, this technique can be extended to identify similarity between natural language requirements in software product lines [25]. Finally, comparing word2vec with other word embedding techniques which can be used for detecting domain specific ambiguities, e.g., GloVe and fastText is also left for future research.

REFERENCES

[1] M. E. C. Hull, K. Jackson, and J. Dick, *Requirements Engineering*, 2nd ed. Springer, 2005.

[2] K. Pohl, *Requirements Engineering - Fundamentals, Principles, and Techniques*, 1st ed. Springer, 2010.

[3] D. M. Berry, E. Kamsties, and M. M. Krieger, "From contract drafting to software specification: Linguistic sources of ambiguity," 2003.

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

[5] B. Gleich, O. Creighton, and L. Kof, "Ambiguity detection: Towards a tool explaining ambiguity sources," in *Requirements Engineering: Foundation for Software Quality (REFSQ)*. Springer Berlin Heidelberg, 2010, pp. 218–232.

[6] S. Gnesi, G. Lami, and G. Trentanni, "An automatic tool for the analysis of natural language requirements," *Comput. Syst. Sci. Eng.*, vol. 20, no. 1, 2005.

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

[8] C. Arora, M. Sabetzadeh, L. C. Briand, and F. Zimmer, "Automated checking of conformance to requirements templates using natural language processing," *IEEE Transactions on Software Engineering*, vol. 41, no. 10, pp. 944–968, 2015.

[9] A. Cimatti, M. Roveri, A. Susi, and S. Tonetta, "Formalizing requirements with object models and temporal constraints," *Software and System Modeling*, vol. 10, no. 2, pp. 147–160, 2011.

[10] L. Kof, "From requirements documents to system models: A tool for interactive semi-automatic translation," in *Proceedings of the 18th IEEE International Requirements Engineering Conference*, September 2010, pp. 391–392.

[11] L. Mich, "On the use of ambiguity measures in requirements analysis," in *Proceedings of the 6th International Workshop on Applications of Natural Language for Information Systems (NLDB)*, June 2001, pp. 143–152.

[12] N. Kiyavitskaya, N. Zeni, L. Mich, and D. M. Berry, "Requirements for tools for ambiguity identification and measurement in natural language requirements specifications," in *Proceedings of the 10th Workshop on Requirements Engineering (WER)*, 2007, pp. 197–206.

[13] N. Kiyavitskaya, N. Zeni, L. Mich, and D. M. Berry", "Requirements for tools for ambiguity identification and measurement in natural language requirements specifications," *Requirements Engineering*, vol. 13, no. 3, pp. 207–239, 2008.

[14] H. Yang, A. N. D. Roeck, V. Gervasi, A. Willis, and B. Nuseibeh, "Extending nocuous ambiguity analysis for anaphora in natural language requirements," in *18th IEEE International Requirements Engineering Conference*, September 2010, pp. 25–34.

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

[16] A. Ferrari, P. Spoletini, and S. Gnesi", "Ambiguity as a resource to disclose tacit knowledge," in *23rd IEEE International Requirements Engineering Conference (RE)*, August 2015, pp. 26–35.

[17] A. Ferrari, P. Spoletini, and S. Gnesi, "Ambiguity and tacit knowledge in requirements elicitation interviews," *Requirements Engineering*, vol. 21, pp. 333–355, 2016.

[18] A. Ferrari and S. Gnesi, "Using collective intelligence to detect pragmatic ambiguities," in *20th IEEE International Requirements Engineering Conference*, September 2012, pp. 191–200.

[19] A. Ferrari, G. Lipari, S. Gnesi, and G. O. Spagnolo, "Pragmatic ambiguity detection in natural language requirements," in *1st IEEE International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*, August 2014, pp. 1–8.

[20] A. Ferrari, B. Donati, and S. Gnesi, "Detecting domain-specific ambiguities: An NLP approach based on wikipedia crawling and word embeddings," in *25th IEEE International Requirements Engineering Conference Workshops (REW)*, September 2017, pp. 393–399.

[21] A. Ferrari, A. Esuli, and S. Gnesi, "Identification of cross-domain ambiguity with language models," in *5th International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*, August 2018, pp. 31–38.

[22] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[23] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'13, 2013, pp. 3111–3119.

[24] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14, 2014, pp. 2177–2185.

[25] K. Pohl, G. Böckle, and F. J. v. d. Linden, *Software Product Line Engineering: Foundations, Principles and Techniques*. Berlin, Heidelberg: Springer-Verlag, 2005.