Cross-Domain Ambiguity Detection using Linear Transformation of Word Embedding Spaces

Vaibhav Jain, Sanskar Jain and Nishant Tanwar

October 27, 2019

Cross-Domain Ambiguity Detection using Linear Transformation of Word Embedding Spaces

Vaibhav Jain, Sanskar Jain, Nishant Tanwar
Delhi Technological University

Abstract

The requirements engineering process is a crucial stage of the software development life cycle. It involves various stakeholders from different professional backgrounds, particularly in the requirements elicitation phase. Each stakeholder carries distinct domain knowledge, causing them to differently interpret certain words, leading to cross-domain ambiguity. This can result in misunderstanding amongst them and jeopardize the entire project. We propose a computationally cheap natural language processing approach to find potentially ambiguous words for a given set of domains. The idea is to apply linear transformations on word embedding models trained on different domain corpora, to bring them into a unified embedding space. We then find words with divergent embeddings as they signify a variation in the meaning across the domains. Applying the approach to a set of hypothetical scenarios produces promising results. It can help a requirements analyst in preventing misunderstandings during elicitation interviews and meetings by defining a set of potentially ambiguous terms in advance.

Keywords: Cross-domain ambiguity, Work embeddings, Linear transformation, Requirements engineering, Natural language processing

Introduction

In the context of software engineering, requirements engineering aims to describe the intended behaviour of a software system along with the associated constraints. It can be viewed in terms of seven phases: inception, elicitation, elaboration, negotiation, specification, validation, and management (Pressman, 2010).

Requirements elicitation has been termed as the most difficult, critical, and communication-intensive aspect of software development (Aggarwal and Singh, 2005). It requires interaction between different stakeholders through various techniques like brainstorming sessions and facilitated application specification technique. A stakeholder is any person with a vested interest in the project, such as potential users, developers, testers, domain experts, and regulatory agency personnel (Singh and Malhotra, 2012). As these stakeholders come from different professional backgrounds and carry different domain knowledge, cross-domain ambiguity can occur amongst them. One may assign an interpretation to another's expression different from the intended meaning. This results in misunderstanding and distrust in requirements elicitation meetings, and costly problems in the later stages of the software life cycle (Wang et al., 2013).

The first attempt to deal with cross-domain ambiguity in requirements engineering was by Ferrari et al. (2017) who used Wikipedia crawling and word embeddings (Mikolov et al., 2013b) to estimate ambiguous computer science (CS) terms vis-à-vis other application domains. Mishra and Sharma (2019) extended this work by focusing on various engineering subdomains. Another approach was suggested by Ferrari et al. (2018) which also considered the ambiguity caused by non-CS domain-specific words and addressed some of the technical limitations of the previous work. This approach was later extended to include quantitative evaluation of the obtained results (Ferrari and Esuli, 2019). An alternative approach which doesn't require domain-specific word embeddings was suggested by Toews and Holland (2019).

In this paper, we propose a natural language processing (NLP) approach based on linear transformation of word embedding spaces. Word embedding is a vector representation of a word capable of capturing its semantic and syntactic relations. A linear transformation can be used to learn a linear relationship between two word embedding spaces. The proposed approach produces a ranked list of potentially ambiguous terms for a given set of domains. We construct a word embedding space for each domain using corpora composed of Wikipedia articles. We then apply linear transformations on these spaces in order to align them and construct a *unified embedding space*. For each word in a set of dominant shared terms, we consider the *domain-specific embeddings* which satisfy a minimum frequency threshold. An ambiguity score is then assigned to the word by applying a distance metric on these embeddings.

The remainder of this paper is organised as follows: We first provide some background on ambiguity in requirements engineering and linear transformation of word embedding space in Section 2. The existing approaches to cross-domain ambiguity detection are briefly explained in Section 3. We outline the proposed approach in Section 4, and the results are presented and discussed in Section 5. Final remarks are provided in Section 6.

Background

Ambiguity in Requirements Engineering

Ambiguity refers to the ability of a natural language (NL) expression to be interpreted in multiple manners. As requirements elicitation is a communication-intensive process, ambiguity is a major negative factor as it can lead to an unclear and incomplete Software Requirements Specification (SRS) document. SRS needs to be unambiguous since it acts as a legal contract between the developers and customers, guides the subsequent development phases and is vital for quality control (de Bruijn and Dekkers, 2010). Most of the existing literature on ambiguity in requirements engineering is focused on written requirement documents, and the role of ambiguity in oral NL during elicitation interviews has not been investigated thoroughly (Ferrari et al., 2016).

Ambiguity can cause *misunderstanding situations* during elicitation interviews, where the requirements analyst does not understand the customer's expression or interprets it incorrectly. The latter phenomenon is known as subconscious disambiguation and is one of the major causes of requirements failure (Gause and Weinberg, 1989). It is difficult to identify unless the interpretation by the analyst is not *acceptable* in his or her mental framework (Ferrari et al., 2016). The problem of cross-domain ambiguity can be seen as a special case of subconscious disambiguation which is caused due to different domain knowledge.

Word Embeddings

Word embedding is a collective term for language modelling techniques that map each word in the vocabulary to a dense vector representation. Contrary to one-hot representation, word embedding techniques embed each word into a low-dimensional continuous space and capture its semantic and syntactic relationships (Li et al., 2015). It is based on the distributional hypothesis proposed by Harris (1954) which states that words appearing in similar linguistic contexts share similar meanings.

One of the most popular word embedding techniques is skip-gram with negative sampling (SGNS) proposed by Mikolov et al. (2013b). It trains a shallow two-layer neural network which, given a single input word w, predicts a set of context words $c(w)$. The context for a word w_i is the set of words surrounding it in a fixed-size window, i.e. $\{w_{i-L}, \cdots, w_{i-1}, w_{i+1}, \cdots, w_{i+L}\}$, where L is the context-window size. Each word w is associated with vectors $u_w \in \mathbb{R}^D$ and $v_w \in \mathbb{R}^D$, called the input and output vectors respectively. If T is the number of windows in the given corpus, then the objective of the skip-gram model is to maximize

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-L\leqslant i\leqslant L; i\neq 0}\log p(w_{t+i}|w_t) \tag{1}$$

If we use the softmax function to define $p(w_{t+i}|w_i)$, then

$$p(w_O|w_I) = \frac{\exp\left(u_{w_I}^T v_{w_O}\right)}{\sum\limits_{w=1}^{W}\exp\left(u_{w_I}^T v_w\right)} \tag{2}$$

where W is the number of words in the vocabulary. However, this formula is not used in practice due to the high computational cost of computing $\Delta p(w_O|w_I)$. An alternative approximation is the negative sampling method, in which the probability function changes to

$$p(w_O|w_I) = \log\sigma(u_{w_I}^T v_{w_O}) + \sum_{i=1}^{k}\log\sigma(-u_{w_I}^T v_{w_i}) \tag{3}$$

where $w_i \sim P(w)$ and $P(w)$ is the noise distribution.

Linear Transformation. A linear transformation can be used to learn a linear mapping from one vector space to another. Its use for combining different word embedding spaces was first explored by Mikolov et al. (2013a) who used it for bilingual machine translation. They used a list of word pairs $\{x_i, y_i\}_{i=1}^{n}$, where y_i is the translation of x_i. Then they learned a *translation matrix* W by minimizing the following loss function

$$\sum_{i=1}^{n}|x_i W - y_i| \tag{4}$$

This approach can also be used for aligning monolingual word embeddings. If we assume that the meaning of most words remains unchanged, linear regression can be used to find the best rotational alignment between two word embedding spaces. Failure to properly align a word can be then used to identify a change in meaning. This is the basis for our approach towards identifying cross-domain ambiguous words. Similar approaches have been used to detect linguistic variation in the meaning of a word with time (Kulkarni et al., 2015; Hamilton et al., 2016) and to develop ensemble word embedding models (Muromägi et al., 2017).

Significant work has been done to improve the linear transformation method. Xing et al. (2015) noticed a hypothetical inconsistency in the distance metrics used in the optimization objectives in Mikolov et al. (2013a): dot product for training word embeddings, Euclidean distance for learning transformation matrix, and cosine distance for similarity computations. It was solved by normalizing the word embeddings and by requiring the transformation matrix to be orthogonal. The optimal orthogonal transformation matrix to map X to Y is given by

$$W = VU^T \tag{5}$$

where $Y^T X = U\Sigma V^T$ is the singular value decomposition (SVD) factorization of $Y^T X$. Dimension-wise mean centering has been shown to improve the performance of linear transformation methods in downstream tasks (Artetxe et al., 2016).

Related Work

Several approaches have been suggested for identification of cross-domain ambiguous words in the context of requirements engineering. The first approach was suggested by Ferrari et al. (2017) who employed Wikipedia crawling and word embeddings to estimate the variation of typical CS words (e.g., code, database, windows) in other domains. They used Wikipedia articles to create two corpora: a CS one and a domain-specific one, replaced the target words (top-k most frequent nouns in the CS corpus) in the latter by a uniquely identifiable modified version, and trained a single language model for both corpora. Cosine similarity was then used as a metric to estimate the variation in the meaning of the target words when they are used in the specified domain. However, this approach suffers from two drawbacks: (a) the inability to identify non-CS cross-domain ambiguous words and (b) the need to construct a language model for each combination of domains. This approach was extended by Mishra and Sharma (2019) who applied it on various subdomains of engineering with varying corpus size. They used the obtained results to identify a similarity threshold for ambiguous words.

Ferrari et al. (2018) suggested an approach based on developing word embedding spaces for each domain, and then estimating the variation in the meaning of a word by comparing the lists of its most similar words in each domain. This approach addressed the above-mentioned drawbacks of the previous one. It was later extended by Ferrari and Esuli (2019), with the major contribution being the introduction of a systematical evaluation of the approach.

An alternative approach which doesn't require domain-specific word embeddings was suggested by Toews and Holland (2019). It estimates a word's similarity across domains through context similarity. This approach does require trained word embeddings, but they are not domain-specific, which allows it to be used on small domain corpora as well. If D_1 and D_2 are two domain corpora, then the context similarity of a word w is defined as

$$simc(w) = \frac{center(c_1) \cdot center(c_2)}{\|center(c_1)\| \cdot \|center(c_2)\|} \tag{6}$$

$$center(c) = \frac{1}{|c|} \sum_{w \in c} IDF_D(w) \cdot v_w \tag{7}$$

where $c_1 \subset D_1$ and $c_2 \subset D_2$ consist of all words from sentences containing w.

Approach

The proposed approach to find a ranked list of potentially ambiguous terms for a given set of domains is depicted in Figure 1.

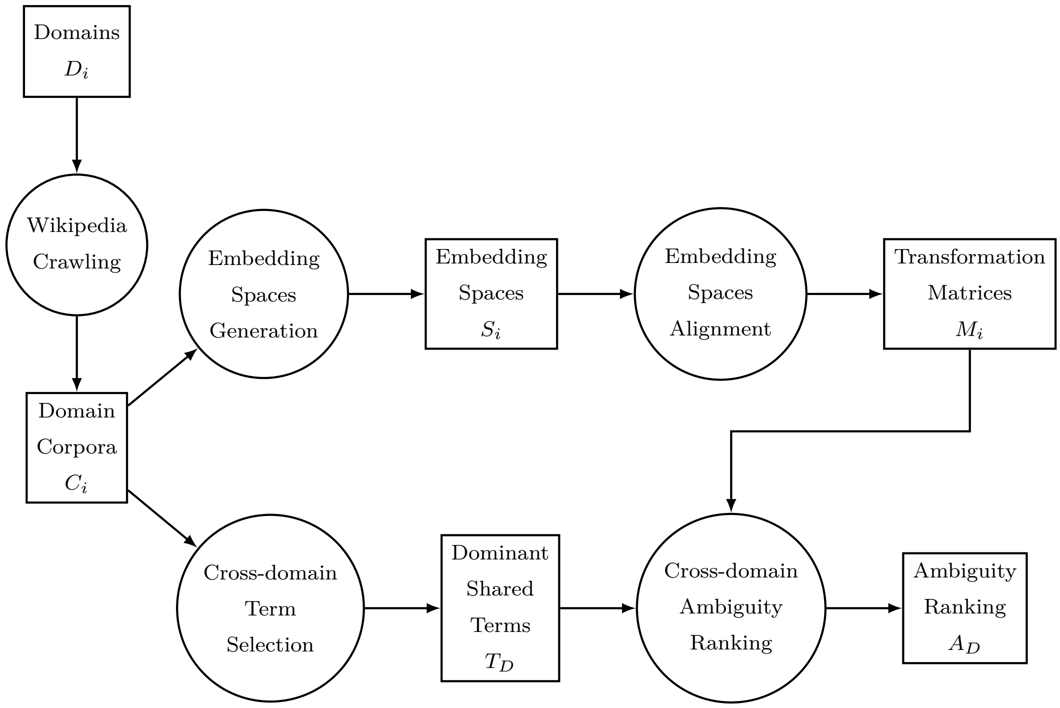

Figure 1. The proposed approach based on word embedding space alignment

Wikipedia Crawling

Given a set of domains $D = \{D_1, \cdots, D_n\}$, this step constructs a set of corpora $C = \{C_1, \cdots, C_n\}$. Each corpus includes Wikipedia articles from a domain-specific category. Each category is a collection of articles and other subcategories. Given the root category, we perform a breadth-first search on its subcategories and add all the reachable articles to the domain corpus. A maximum subcategory depth of 3 is kept during the search to ensure that only relevant articles are added, and the number of total articles in a corpus is also limited to 20,000 to avoid corpus size imbalance across domains.

Corpus generation is followed by *text preprocessing* – an auxiliary step not depicted in Figure 1. This involves (a) converting each word to lowercase, (b) stop-words removal, and (c) lemmatization. Stop-words refer to common words that appear quite frequently in a natural language and are removed before various NLP tasks. Lemmatization refers to the process of reducing a word from an inflectional form to its lemma, i.e. root word.

Embedding Spaces Generation

A word embedding space S_i is generated for each corpus $C_i \in C$. The SGNS variant of the `word2vec` algorithm (Mikolov et al., 2013b) is used to train the word embeddings. This involves defining the dimension of the word embeddings d, context window size L, the number of noise samples η, and the minimum frequency f_{min} for a word to be considered. This step is followed by an auxiliary step called *embedding preprocessing*, which consists of length normalization and dimension-wise mean centring.

Embedding Spaces Alignment

This step determines a transformation matrix M_i for each domain D_i which maps it to a unified embedding space. The algorithm for this step is reported as Algorithm 1.

Algorithm 1 Aligning word embedding spaces

procedure $\textsc{AlignWordEmbeddings}(S)$
  $M_1 \leftarrow I_d$
  $S' \leftarrow \{S_1\}$
  for $S_i \in S \setminus S_1$ **do**
    $X, Y \leftarrow []$
    for $w_j \in \textsc{Vocabulary}(S_i)$ **do**
      $V \leftarrow \emptyset$
      for $S_k \in S'$ **do**
        if $w_j \in \textsc{Vocabulary}(S_k)$ **then**
          $V \leftarrow V \cup \{S_k(w_j) \cdot M_k\}$
      if $V \neq \emptyset$ **then**
        $X.\text{insert}(S_i(w_j))$
        $Y.\text{insert}(\textsc{Average}(V))$
    $U, \Sigma, V^T \leftarrow \text{SVD}(YX^T)$
    $M_i \leftarrow VU^T$
  return M

The transformation matrices are determined incrementally. The transformation matrix M_1 for S_1 is the identity matrix I_d. Subsequent M_i maps the corresponding S_i to the average of the transformed versions of its previous embedding spaces $S_1, S_2, \cdots, S_{i-1}$. More specifically, for each word w_j in the vocabulary of S_i, the target vector y_{ij} is defined as the average of the corresponding word embeddings in the already transformed spaces. These pairs $(S_i(w_j), y_{ij})$ are then used to learn the optimal transformation matrix M_i which is constrained to be orthogonal.

Cross-Domain Term Selection

The approach for identifying dominant shared terms T_D has been reported as Algorithm 2.

Algorithm 2 Selecting dominant shared terms

procedure $\textsc{SelectTerms}(C, k, \rho)$
  $T_D \leftarrow \emptyset$
  for $w_i \in \textsc{Vocabulary}(C_1) \cup \cdots \cup \textsc{Vocabulary}(C_n)$ **do**
    if $\text{POS}(w_i) = \text{NN}$ **then**
      $counts = \{\textsc{Freq}(C_1, w_i), \cdots, \textsc{Freq}(C_n, w_i)\}$
      $c_1, c_2 \leftarrow \textsc{Top2Values}(counts)$
      if $c_1 \geqslant p \wedge c_2 \geqslant \rho \times c_1$ **then**
        $T_D \leftarrow T_D \cup \{w_i\}$
  return T_D

This step requires two numerical parameters, k and ρ. To be considered a dominant shared term, a word w must satisfy two conditions:

1. Its maximum frequency in a domain corpus, i.e. $f_{max} = max(count_i(w))$, should be greater than or equal to k.

2. It should have a frequency of at least ρf_{max} in any other domain corpus.

We limited the scope to only nouns, but this approach can be extended to other parts of speech as well.

Cross-Domain Ambiguity Ranking

This step assigns an *ambiguity score* to each word in T_D based on their cross-domain ambiguity across the corpora $C = \{C_1, \cdots, C_n\}$. The algorithm for the same is reported as Algorithm 3.

Algorithm 3 Assigning ambiguity scores

procedure $\textsc{AssignAmbiguityScores}(T_D, M, S)$
 $Score \leftarrow \emptyset$
 for $w \in T_D$ **do**
  $V \leftarrow \emptyset$
  $counts = \{\textsc{Freq}(S_1, w), \cdots, \textsc{Freq}(S_n, w)\}$
  $f_{max} \leftarrow \textsc{Max}(counts)$
  for $S_i \in S$ **do**
   if $w \in S_i$ **then**
    $V \leftarrow V \cup \{M_i S_i(w)\}$
  $U \leftarrow 0$
  for $v_i \in V$ **do**
   for $v_j \in V \setminus v_i$ **do**
    $U \leftarrow U + \textsc{CosineDistance}(v_i, v_j)$
  $Score[w] \leftarrow \frac{U}{|V|(|V|-1)}$
 $A_D \leftarrow \textsc{Sort}(T_D, Score)$
 return A_D

The idea is as follows. For each word w in the set of dominant shared terms T_D, we find the cosine distance for each unordered pair of its transformed embeddings, which is given by

$$cosineDistance(v_i, v_j) = 1 - cosineSimilarity(v_i, v_j) \tag{8}$$

$$cosineSimilarity(v_i, v_j) = \frac{v_i \cdot v_j}{\|v_i\|\|v_j\|} \tag{9}$$

The average of all these cosine distances is the ambiguity score assigned to the word w. All words in T_D are sorted according to their score and a ranked list A_D is produced.

Our linear transformation-based approach is computationally cheaper than the one suggested by Ferrari and Esuli (2019) which relies on k-nearest neighbour (KNN) search. We also hypothesize that it is logically more sound: the KNN approach assumes that the meanings of the neighbouring words remain constant across domains, but this assumption fails for *ambiguous clusters*. For example, a lot of topics in artificial intelligence, such as neural networks and genetic algorithms, are inspired by biology. Due to this, certain words appear together in both these domains but carry different interpretations. On the other hand, our approach works on a much weaker assumption that the meaning of most words remains the same across domains. Hence, it judges a word's meaning from a global context rather than a local one.

Results

Project Scenarios

To showcase the working of our approach, we have considered the same hypothetical project scenarios similar that were used by Ferrari and Esuli (2019). They involve five domains: computer science (CS), electronic engineering (EE), mechanical engineering (ME), medicine (MED), and sports (SPO).

1. *Light Controller* [CS, EE]: an embedded software for room illumination system
2. *Mechanical CAD* [CS, ME]: a software for designing and drafting mechanical components.
3. *Medical Software* [CS, MED]: a disease-prediction software.
4. *Athletes Network* [CS, SPO]: a social network for athletes.
5. *Medical Device* [CS, EE, MED]: a fitness tracker connected to a mobile app
6. *Medical Robot* [CS, EE, ME, MED]: a computer-controlled robotic arm used for surgery.
7. *Sport Rehab Machine* [CS, EE, ME, MED, SPO]: a rehabilitation machine targeted towards athletes.

Experimental Setup

We used the Wikipedia API for Python[1] to create domain corpora. A maximum subcategory depth of 3 and a maximum article limit of 20,000 is set while creating each domain corpus.[2] We converted each article text to lowercase and removed all non-alphanumeric words. The article count, word count, and vocabulary size for each domain corpus is reported by Table 1.

Table 1
Domain corpora statistics

Domain	Articles	Words	Vocabulary
Computer science	20,000	80,37,521	1,77,764
Electronic engineering	16,420	77,10,843	1,79,898
Mechanical engineering	20,000	1,02,02,205	1,99,696
Medicine	20,000	80,45,379	2,00,266
Sports	20,000	94,48,453	2,42,583

The word embeddings were trained using the `gensim`[3] implementation of the `word2vec` SGNS algorithm with word embedding dimension $d = 50$, context window size $L = 10$, negative sampling size $\eta = 5$, and minimum frequency $f_{min} = 10$. For identifying dominant shared terms, the parameters were set as $k = 1000$ and $\rho = 0.3$.

[1] https://pypi.org/project/wikipedia/

[2] Since Category:Computer science is a subcategory of Category:Electronic engineering, it was excluded while creating the EE corpus to avoid extensive overlap with the CS corpus.

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

Cross-Domain Ambiguity Rankings

We have reported the top-20 and bottom-20 ranked terms for each project scenario along with their ambiguity scores in Tables 2, 3, and 4 (terms referred in the text have been highlighted). Here, we discuss some of the notable inferences that can be made from these results.

1. We can observe that the ambiguity scores for the light controller scenario are considerably lower than that for other scenarios. This is on expected lines as CS and electronic engineering are closely related fields. The ambiguity seems to increase as we go from technical domains like mechanical engineering and medicine to a non-technical one like sports. This shows that our approach can be used to quantitatively define the technical similarity between domains.

2. Most of the high ranked words in the light controller [CS, EE] scenario seem to be of technical nature, with an important exception being the word *family*. By inspecting a word's nearest neighbours in individual domain-specific embedding spaces, we can estimate its meaning in those domains. The word *family* is nearest to *mildly*, *immigrated*, and *grzegorczyk* in the CS domain, and to *6100*, *8051*, and *7400* in the EE domain. This indicates that it is mostly used to denote an electronic component series in the EE domain. The nearest words for *translation* are *translator*, *translate*, and *nlp* in CS, and *syntax*, *semantics*, and *parsing* in EE. This is a case of pragmatic ambiguity as the word can be used in an NLP or a compiler design context.

3. The word *assembly* occurs amongst the top-20 ranked terms in all scenarios except light controller and sports rehab machine. Its nearest words in each domain are (a) CS: *assembler*, *compiler*, *verilog*, (b) EE: *assembling*, *tooling*, *assembled*, (c) ME: *joint*, *assembled*, *housing*, (d) MED: *election*, *wha*, *vote*, and (e) SPO: *congress*, *cgf*, *legislative*.

4. The most ambiguous word for the set of all five domains is *induction* with an ambiguity score of 0.8868. Its nearest words in each domain are (a) CS: *inductive*, *equational*, *deduction*, (b) EE: *dynamo*, *inductive*, *emf*, (c) ME: *homopolar*, *reluctance*, *magnetizing*, (d) MED: *inducing*, *initiation*, *suppression*, and (e) SPO: *inductee*, *inducting*, *honoree*.

5. Most of the low-ranked terms are either generic terms such as *infrastructure*, *opportunity*, and *nature*, or common names such as *robert* and *peter*. The word *government*, with an ambiguity score of 0.1438, is the least ambiguous term for the set of all domains. This fact can be verified by looking at its nearest words: (a) CS: *governmental*, *ministry*, *federal*, (b) EE: *mandate*, *policy*, *legislation*, (c) ME: *authority*, *immigration*, *diplomatic*, (d) MED: *authority*, *governmental*, *obligation*, and (e) SPO: *authority*, *policy*, *governmental*.

Table 2
Ranked list of dominant shared terms for project scenarios (a) light controller and (b) mechanical CAD.

Light controller [CS, EE]		Mechanical CAD [CS, ME]	
Term	Score	Term	Score
family	0.6661	thread	1.0477
translation	0.6505	**induction**	0.8943
deal	0.6401	lighting	0.8412
base	0.6390	freedom	0.8011
weight	0.6287	race	0.7890
kingdom	0.6229	sun	0.7652
derivative	0.6202	**assembly**	0.7277
box	0.6179	background	0.7152
grid	0.6053	separation	0.7107
volume	0.5890	translation	0.7098
gap	0.5786	compression	0.6950
chain	0.5693	machinery	0.6819
mark	0.5401	base	0.6807
differential	0.5401	weight	0.6770
generator	0.5385	root	0.6724
actor	0.5236	profile	0.6666
curve	0.5195	trace	0.6334
studio	0.5136	utility	0.6329
press	0.5084	fiber	0.6288
expansion	0.5071	disc	0.6272
⋮	⋮	⋮	⋮
reason	0.1246	threat	0.1439
community	0.1228	**nature**	0.1438
innovation	0.1227	episode	0.1431
benefit	0.1222	fact	0.1418
support	0.1203	cost	0.1390
monitoring	0.1203	technique	0.1373
app	0.1180	technology	0.1349
robert	0.1177	mother	0.1324
ceo	0.1175	commission	0.1320
email	0.1174	daughter	0.1312
instruction	0.1166	**infrastructure**	0.1311
bandwidth	0.1147	benefit	0.1297
country	0.1125	step	0.1276
company	0.1118	situation	0.1204
authority	0.1060	notion	0.1157
photo	0.1035	lack	0.1142
lack	0.0977	authority	0.1024
founder	0.0961	wife	0.0982
infrastructure	0.0960	story	0.0954
government	0.0893	**government**	0.0929

Table 3
Ranked list of dominant shared terms for project scenarios (a) medical software and (b) athletes network.

Medical software [CS,MED]		Athletes network [CS, SPO]	
Term	Score	Term	Score
mouse	1.0655	**assembly**	1.0774
assembly	0.9194	advance	1.0380
compression	0.8661	receiver	0.9631
derivative	0.8546	goal	0.9410
conversion	0.8535	field	0.9347
agent	0.8072	uniform	0.9115
root	0.7846	delivery	0.9032
sun	0.7828	touch	0.8898
base	0.7719	tag	0.8813
kingdom	0.7641	sun	0.8545
branch	0.7503	gear	0.8487
expression	0.7446	boot	0.8298
domain	0.7385	engine	0.8157
mass	0.7375	bell	0.8046
background	0.7274	gate	0.7916
column	0.7180	ring	0.7880
line	0.7166	appearance	0.7879
scale	0.6996	sign	0.7819
case	0.6720	capture	0.7723
host	0.6709	balance	0.7672
⋮	⋮	⋮	⋮
angle	0.1715	treatment	0.1926
demand	0.1708	temperature	0.1925
institution	0.1700	audience	0.1898
shape	0.1657	consequence	0.1872
experiment	0.1650	evidence	0.1861
campaign	0.1634	entertainment	0.1818
decade	0.1629	hospital	0.1817
understanding	0.1564	education	0.1798
topic	0.1564	authority	0.1782
government	0.1531	damage	0.1738
discussion	0.1513	culture	0.1730
report	0.1427	movie	0.1708
article	0.1413	army	0.1706
nature	0.1384	report	0.1621
peter	0.1367	sale	0.1599
robert	0.1277	interview	0.1556
publication	0.1257	benefit	0.1529
opportunity	0.1256	**robert**	0.1501
thomas	0.1247	market	0.1428
education	0.1190	**government**	0.1167

Table 4
Ranked list of dominant shared terms for project scenarios (a) medical device, (b) medical robot, and (c) sports rehab machine.

Medical device [CS, EE, MED]		Medical robot [CS, EE, ME, MED]		Sport rehab machine [CS, EE, ME, MED, SPO]	
Term	Score	Term	Score	Term	Score
analogue	0.8034	stroke	0.8567	**induction**	0.8868
kernel	0.8014	thread	0.8334	partition	0.8366
valve	0.7802	spark	0.7612	stroke	0.8129
mouse	0.7706	injection	0.7534	thread	0.8057
driver	0.7706	**induction**	0.7328	pipeline	0.7878
packet	0.7502	pipeline	0.7259	suspension	0.7830
processor	0.7475	sugar	0.7192	root	0.7804
mac	0.7475	analogue	0.7176	toe	0.7782
gate	0.7441	trace	0.7103	hammer	0.7669
root	0.7375	root	0.7067	trace	0.7559
calculator	0.7333	mouse	0.6855	mouse	0.7438
thread	0.7331	**assembly**	0.6841	relay	0.7433
assembly	0.7164	compression	0.6784	rifle	0.7411
resistance	0.7145	strain	0.6763	ice	0.7368
window	0.7068	expansion	0.6748	analogue	0.7281
intel	0.7049	cd	0.6652	pistol	0.7265
kingdom	0.6990	oracle	0.6603	mac	0.7243
bit	0.6884	kingdom	0.6597	glider	0.7183
pipeline	0.6841	clock	0.6539	lane	0.7170
iron	0.6835	grid	0.6520	gate	0.7166
⋮	⋮	⋮	⋮	⋮	⋮
infrastructure	0.1841	company	0.1782	report	0.2011
country	0.1782	founder	0.1781	policy	0.2007
shape	0.1756	**nature**	0.1774	money	0.1990
nature	0.1738	understanding	0.1743	subsidiary	0.1963
discussion	0.1732	country	0.1719	**peter**	0.1957
institution	0.1730	subsidiary	0.1716	circumstance	0.1953
company	0.1711	**infrastructure**	0.1705	father	0.1936
publication	0.1706	publication	0.1703	**robert**	0.1928
committee	0.1659	shape	0.1687	decade	0.1889
founder	0.1637	purchase	0.1662	**infrastructure**	0.1871
compiler	0.1607	decade	0.1626	**opportunity**	0.1860
thomas	0.1520	chairman	0.1605	fear	0.1842
opportunity	0.1497	town	0.1573	daniel	0.1809
authority	0.1444	**opportunity**	0.1548	understanding	0.1784
decade	0.1444	joseph	0.1521	purchase	0.1734
partnership	0.1431	**peter**	0.1470	wife	0.1708
byte	0.1422	money	0.1458	joseph	0.1648
peter	0.1412	authority	0.1305	love	0.1644
robert	0.1326	**government**	0.1292	authority	0.1483
government	0.1256	wife	0.1232	**government**	0.1438

Conclusion and Future Work

Ambiguous requirements are a major hindrance to successful software development and it is necessary to avoid them from the elicitation phase itself. Although this problem has been studied extensively, cross-domain ambiguity has attracted research only in recent times. We have proposed a novel approach which makes use of linear transformation to map various domain-specific language models into a unified embedding space, allowing comparison of word embeddings trained from different corpora. Our work provides a computationally efficient way of determining potentially ambiguous words. The planned future work includes (a) quantitative evaluation, (b) experiments with distance metrics other than average pairwise cosine distance, (c) defining an ambiguity threshold, and (d) identifying better corpora sources.

References

Aggarwal, K. and Singh, Y. (2005). *Software Engineering*. New Age International (P) Limited.

Artetxe, M., Labaka, G., and Agirre, E. (2016). Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294, Austin, Texas. Association for Computational Linguistics.

de Bruijn, F. and Dekkers, H. L. (2010). Ambiguity in natural language software requirements: A case study. In *Requirements Engineering: Foundation for Software Quality*, pages 233–247, Berlin, Heidelberg. Springer Berlin Heidelberg.

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

Ferrari, A. and Esuli, A. (2019). An NLP approach for cross-domain ambiguity detection in requirements engineering. *Automated Software Engineering*, 26(3):559–598.

Ferrari, A., Esuli, A., and Gnesi, S. (2018). Identification of cross-domain ambiguity with language models. In *2018 5th International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*, pages 31–38.

Ferrari, A., Spoletini, P., and Gnesi, S. (2016). Ambiguity and tacit knowledge in requirements elicitation interviews. *Requirements Engineering*, 21(3):333–355.

Gause, D. C. and Weinberg, G. M. (1989). *Exploring Requirements: Quality Before Design*. Dorset House Publishing Co., Inc., New York, NY, USA.

Hamilton, W. L., Leskovec, J., and Jurafsky, D. (2016). Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1489–1501, Berlin, Germany. Association for Computational Linguistics.

Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3):146–162.

Kulkarni, V., Al-Rfou, R., Perozzi, B., and Skiena, S. (2015). Statistically significant detection of linguistic change. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 625–635, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Li, Y., Xu, L., Tian, F., Jiang, L., Zhong, X., and Chen, E. (2015). Word embedding revisited: A new representation learning and explicit matrix factorization perspective. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 3650–3656. AAAI Press.

Mikolov, T., Le, Q. V., and Sutskever, I. (2013a). Exploiting similarities among languages for machine translation. *ArXiv*, abs/1309.4168.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

Mishra, S. and Sharma, A. (2019). On the use of word embeddings for identifying domain specific ambiguities in requirements. In *Proceedings of the 27th International Requirements Engineering Conference Workshops (REW)*, pages 234–240. IEEE Computer Society.

Muromägi, A., Sirts, K., and Laur, S. (2017). Linear ensembles of word embedding models. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pages 96–104, Gothenburg, Sweden. Association for Computational Linguistics.

Pressman, R. S. (2010). *Software Engineering: A Practitioner's Approach*. McGraw-Hill.

Singh, Y. and Malhotra, R. (2012). *Object-Oriented Software Engineering*. PHI Learning.

Toews, D. and Holland, L. V. (2019). Determining domain-specific differences of polysemous words using context information. In *Joint Proceedings of REFSQ-2019 Workshops, Doctoral Symposium, Live Studies Track, and Poster Track co-located with the 25th International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2019), Essen, Germany, March 18th, 2019.*

Wang, Y., Manotas Gutièrrez, I. L., Winbladh, K., and Fang, H. (2013). Automatic detection of ambiguous terminology for software requirements. In *Natural Language Processing and Information Systems*, pages 25–37, Berlin, Heidelberg. Springer Berlin Heidelberg.

Xing, C., Wang, D., Liu, C., and Lin, Y. (2015). Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011, Denver, Colorado. Association for Computational Linguistics.