EasyChair Preprint

№ 1785

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

Vaibhav Jain, Sanskar Jain and Nishant Tanwar

January 27, 2020

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

Vaibhav Jain*†, Sanskar Jain*‡, and Nishant Tanwar*§
*Department of Software Engineering
Delhi Technological University, Delhi, India
†Email: vaibhav29498@gmail.com
‡Email: sanskar27jain@gmail.com
§Email: nishant.tanwar08@gmail.com

***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. This paper proposes a 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*. The approach then finds words with divergent embeddings as they signify a variation in the meaning across the domains. It can help a requirements analyst in preventing misunderstandings during elicitation interviews and meetings by defining a set of potentially ambiguous terms in advance. The paper also discusses certain problems with the existing approaches and discusses how the proposed approach resolves them.**

***Index Terms*—cross-domain ambiguity, linear transformation, requirements engineering, word embeddings, natural language processing**

I. Introduction

In the context of software engineering, requirements engineering is the process of describing the intended behaviour of a software system along with the associated constraints [1]. One of its phase is requirements elicitation, which has been termed as the most difficult, critical, and communication-intensive aspect of software development [2]. 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 [3]. 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 [4].

The study of variation of word meanings across domains as an NLP problem is termed as *Synchronic Lexical Semantic Change (LSC) Detection* [5]. The first attempt to apply it for dealing with cross-domain ambiguity in requirements engineering was by Ferrari *et al.* (2017) who used Wikipedia crawling and word embeddings to estimate ambiguous computer science (CS) terms vis-à-vis other application domains [6]. Mishra and Sharma extended this work by focusing on various engineering subdomains [7]. 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 [8]. This approach was later extended to include quantitative evaluation of the obtained results [9]. An alternative approach which doesn't require domain-specific word embeddings was suggested by Toews and Holland [10].

This paper proposes 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. It constructs a word embedding space for each domain using corpora composed of Wikipedia articles. It then applies linear transformations on these spaces in order to align them and construct a *unified embedding space*. For each word in a set of target words, an ambiguity score is assigned by applying a distance metric on its *domain-specific embeddings*.

The remainder of this paper is organised as follows: Section II provides some background on word embeddings and linear transformation of embedding spaces. The existing approaches to cross-domain ambiguity detection are briefly explained in Section III. The motivation behind the proposed approach is discussed in Section IV, whereas the apporach itself is outlined in Section V. The experimental setup and results are presented and discussed in Section VI, and the conclusion and details of planned future work are provided in Section VII.

II. Preliminaries

A. 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 [11]. It is based on the distributional hypothesis proposed by Harris which states that words appearing in similar linguistic contexts share similar meanings [12].

One of the most popular word embedding techniques is skip-gram with negative sampling (SGNS) [13]. 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

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

In the negative sampling method, $p(w_{t+i}|w_i)$ is defined as

$$p(w_O|w_I) = \log\sigma(u_{w_I}^T v_{w_O}) + \sum_{i=1}^{k}\log\sigma(-u_{w_I}^T v_{w_i}) \tag{2}$$

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

B. 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.* who used it for bilingual machine translation [14]. 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

$$\sum_{i=1}^{n}|x_i W - y_i| \tag{3}$$

This approach can also be used for aligning monolingual word embeddings. If one assumes 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 the proposed approach towards identifying cross-domain ambiguous words. Similar approaches have been used to detect linguistic variation in the meaning of a word with time and to develop ensemble word embedding models [15]–[17].

Significant work has been done to improve the linear transformation method. Dimension-wise mean centering has been shown to improve the performance of linear transformation methods in downstream tasks [18]. Xing *et al.* noticed a hypothetical inconsistency in the distance metrics used in the optimization objectives in the work of Mikolov *et al.*: dot product for training word embeddings, Euclidean distance for learning transformation matrix, and cosine distance for similarity computations [19]. It was solved by normalizing the word embeddings and by requiring the transformation matrix to be orthogonal. The optimal orthogonal transformation matrix which maps X to Y can be found through the solution of the well-known Orthogonal Procrustes problem, which is given by

$$W = UV^T \tag{4}$$

where $X^TY = U\Sigma V^T$ is the singular value decomposition (SVD) factorization of X^TY [20].

III. Related Work

Synchronic LSC detection refers to the measurement of variation of word meanings across domains or speaker communities [5]. The latter has been studied by making use of the large-scale data provided by communities on online platforms such as Reddit [21].

Research works on cross-domain ambiguity detection have been limited to its applicability in 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 [6]. 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 the following drawbacks:

- the inability to identify non-CS cross-domain ambiguous words,
- the need to construct a language model for each combination of domains, and
- the need to modify the domain-specific corpus.

Their work was extended by Mishra and Sharma who applied it on various subdomains of engineering with varying corpus size [7]. They identified the most suitable hyperparameters for training word embeddings on corpora of three different classes: large, medium, and small, based on the number of documents. They then 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 [8]. This approach addressed the above-mentioned drawbacks of the previous one. It was later extended by Ferrari and Esuli, with the major contribution being the introduction of a quantitative evaluation of the approach [9].

An alternative approach which does not require domain-specific word embeddings was suggested by Toews and Holland [10]. It estimates a word's similarity across domains through context similarity. This approach does require trained word embeddings, but they are not required to be 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

$$simc(w) = \frac{center(c_1) \cdot center(c_2)}{\|center(c_1)\| \cdot \|center(c_2)\|} \tag{5}$$

$$center(c) = \frac{1}{|c|} \sum_{w \in c} IDF_D(w) \cdot v_w \tag{6}$$

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

IV. Motivation

The motivation behind the proposed linear transformation based approach is based on the following factors:

- The approaches suggested by Ferrari *et al.* (2018) and Toews and Holland judge a word's meaning from its local context rather than a global one. This leads them to wrongly assign a high ambiguity score to a word having distinct, yet similar, nearest words in different domains. A particular example of this problem is the high score assigned to proper names such as *Michael*; although they are near to other proper nouns in all domains, but the exact lists vary widely. Such approaches also fail in the opposite scenario in which the meaning of the nearest words themselves change. This can happen in the case of *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. However, the approach proposed by this paper relies on the global context rather than the local one, which resolves the issues mentioned above.
- The proposed approach can work for more than two domains as opposed to the approaches suggested by Ferrari *et al.* (2017) and Toews and Holland [6], [10].
- The approach proposed by Ferrari *et al.* (2018) assumes the meaning of the neighbouring words to be the same across domains, whereas a linear transformation based approach works on a much weaker assumption that the meaning of most words remains the same across domains.
- Schlechtweg *et al.* evaluated various synchronic LSC detection models on *SURel*, a German dataset consisting of the meaning variations from general to domain-specific corpus determined through manual annotation [5], [22]. Their study found linear transformation to perform much better than other alignment techniques such as word injection (proposed by Ferrari *et al.* (2017)) and vector initialization.

V. Approach

Given a set of domains $D = \{D_1, \cdots, D_n\}$, the approach requires a word embedding space S_i corresponding to each domain D_i. The first step is to align the embedding spaces (subsection V-A) and then determine the set of target words (subsection V-B). The final step is to assign a cross-domain ambiguity score to each target word (subsection V-C).

A. Embedding Spaces Alignment

This step determines a transformation matrix M_i for each domain-specific word embedding space S_i which maps it to a unified embedding space. It uses an algorithm devised by Muromägi *et al.* [17] which iteratively finds the transformation matrices $M_1, M_2, \cdots, M_n$ and the common target space Y. It performs the following two steps in each iteration:

1) The transformation matrices $M_1, M_2, \cdots, M_n$ are calculated using equation 4.
2) The target space is updated to be the average of all transformed spaces:

$$Y(w) = \frac{1}{n_w} \sum_{i=1}^{n_w} S_i(w) M_i \tag{7}$$

where n_w is the number of domain-specific embedding spaces with word w as part of its vocabulary.

These steps are repeatedly performed as long as the change in average normalised residual error, which is given by

$$\frac{1}{n} \sum_{i=1}^{n} \frac{\|S_i M_i - Y\|}{\sqrt{|S_i| \cdot d}} \tag{8}$$

is equal to or greater than a predefined threshold τ.

B. Target Words Selection

The approach for identifying the set of target words T_D has been presented in Figure 1.

procedure SELECTWORDS(C, k, ρ)
  $T_D \leftarrow \emptyset$
  for $w_i \in \text{VOCAB}(C_1) \cup \cdots \cup \text{VOCAB}(C_n)$ **do**
    if $\text{POS}(w_i) \in \{NN, VB, ADJ\}$ **then**
      $counts = \{\text{FREQ}(C_1, w_i), \cdots, \text{FREQ}(C_n, w_i)\}$
      $c_1, c_2 \leftarrow \text{TOP2VALUES}(counts)$
      if $c_1 \geqslant k \wedge c_2 \geqslant \rho \times c_1$ **then**
        $T_D \leftarrow T_D \cup \{w_i\}$
  return T_D

Fig. 1. Algorithm for selecting target words

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

1) It must be a content word, i.e. noun, verb, or adjective.
2) Its maximum frequency in a domain corpus, i.e. $f_{max} = max(count_i(w))$, should be greater than or equal to k.
3) It should have a frequency of at least ρf_{max} in any other domain corpus.

C. 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 in Figure 2.

The idea is as follows. For each word w in the set of target words T_D, the cosine distance for each unordered pair of its transformed embeddings is calculated, which is given by

procedure ASSIGNAMBIGUITYSCORES(T_D, M, S)
  $Score \leftarrow \emptyset$
  for $w \in T_D$ **do**
    $V \leftarrow \emptyset$
    for $S_i \in S$ **do**
      if $w \in S_i$ **then**
        $V \leftarrow V \cup \{M_i S_i(w)\}$
    $U \leftarrow 0$
    $C \leftarrow 0$
    for $v_i \in V$ **do**
      for $v_j \in V \setminus v_i$ **do**
        $c \leftarrow count_i(w) + count_j(w)$
        $U \leftarrow U + c \times$ COSINEDISTANCE(v_i, v_j)
        $C \leftarrow C + c$
    $Score[w] \leftarrow U/C$
  $A_D \leftarrow$ SORT$(T_D, Score)$
  return A_D

Fig. 2. Algorithm for assigning ambiguity scores

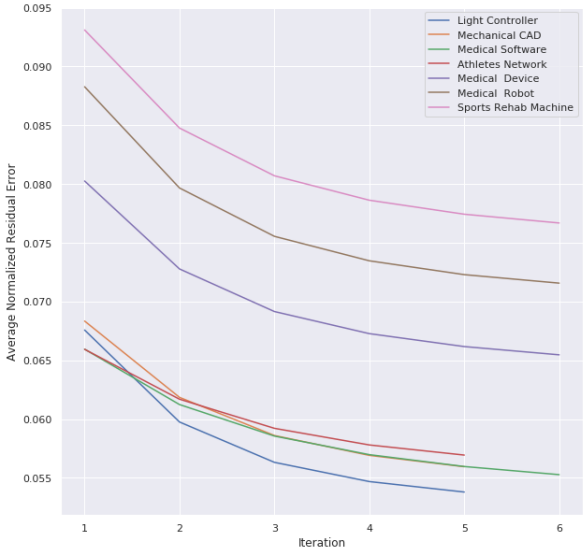

Fig. 3. Plot of the average normalized residual errors

$$cosineDistance(v_i, v_j) = 1 - \frac{v_i \cdot v_j}{\|v_i\|\|v_j\|} \tag{9}$$

The average of all these cosine distances, weighted by the sum of the word frequencies in the corresponding domain corpora, 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.

VI. RESULTS

A. Project Scenarios

To showcase the working of the proposed approach, this paper considers the same hypothetical project scenarios that were used by Ferrari and Esuli [9]. 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) *Sports Rehab Machine* [CS, EE, ME, MED, SPO]: a rehabilitation machine targeted towards athletes.

The first four scenarios can be thought of as an interview between a requirements analyst with a CS and a domain expert, whereas the other three scenarios can be regarded as group elicitation meetings involving stakeholders from multiple domains.

B. Experimental Setup

The Wikipedia API for Python[1] was used to construct the domain corpora by scraping articles belonging to particular categories. A maximum subcategory depth of 3 and a maximum article limit of 20,000 was set while creating each domain corpus.[2] Each article text was converted to lowercase and all non-alphanumeric words and stop words were removed, followed by lemmatization. The article count, word count, and vocabulary size for each domain corpus are reported by Table I.

TABLE I
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 $f_{min} = 10$. Training of the word embeddings was followed by length normalization and dimension-wise mean centering. For aligning the word embedding spaces, the threshold τ was set to 0.001. The plot of average normalized residual errors for each project scenario is depicted in Figure 3. The parameters for identifying target words were set as $k = 1000$ and $\rho = 0.5$.

[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/

C. Cross-Domain Ambiguity Rankings

The top-10 and bottom-10 ranked terms for each project scenario are reported along with their ambiguity scores by Table II.

In order to study the cases of disagreement between the approaches proposed by this paper and Ferrari *et al.* (2018), the top-5 words with the largest absolute differences between the assigned ranks have been reported for each scenario by Table III. The number of target words for each project scenario have also been mentioned in parenthesis. It can be observed that most of the cases of disagreement have a higher rank, i.e. relatively lower ambiguity score assigned by the linear transformation approach proposed by this paper. Most of such cases are proper names such as *robert*, *peter*, and *daniel*. This is because of the problems associated with local-context approaches discussed in Section IV, and the low ambiguity score given to such words by the proposed approach is in line with the expected behavior of a global-context approach.

VII. 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. This paper proposes a global-context 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. It provides a logically effective way of determining potentially ambiguous words and a qualitative comparison with an existing local-context approach produces promising results. The planned future work includes a systematic quantitative evaluation of the proposed approach, extending the approach to consider multi-word phrases, defining an ambiguity threshold, and identifying better corpora sources.

References

[1] R. S. Pressman, *Software Engineering: A Practitioner's Approach*. McGraw-Hill, 2010.

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

[3] Y. Singh and R. Malhotra, *Object-Oriented Software Engineering*. PHI Learning, 2012.

[4] Y. Wang, I. L. Manotas Gutièrrez, K. Winbladh, and H. Fang, "Automatic detection of ambiguous terminology for software requirements," in *Natural Language Processing and Information Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 25–37.

[5] D. Schlechtweg, A. Hätty, M. Del Tredici, and S. Schulte Im Walde, "A wind of change: Detecting and evaluating lexical semantic change across times and domains," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 01 2019, pp. 732–746.

[6] 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)*, Sep. 2017, pp. 393–399.

[7] S. Mishra and A. Sharma, "On the use of word embeddings for identifying domain specific ambiguities in requirements," in *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*, Sep. 2019, pp. 234–240.

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

[9] A. Ferrari and A. Esuli, "An NLP approach for cross-domain ambiguity detection in requirements engineering," *Automated Software Engineering*, vol. 26, no. 3, pp. 559–598, Sep 2019.

[10] D. Toews and L. V. Holland, "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.*, 2019. [Online]. Available: http://ceur-ws.org/Vol-2376/NLP4RE19_paper02.pdf

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

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

[13] 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. USA: Curran Associates Inc., 2013, pp. 3111–3119.

[14] T. Mikolov, Q. V. Le, and I. Sutskever, "Exploiting similarities among languages for machine translation," *ArXiv*, vol. abs/1309.4168, 2013. [Online]. Available: http://arxiv.org/abs/1309.4168

[15] W. L. Hamilton, J. Leskovec, and D. Jurafsky, "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)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1489–1501.

[16] V. Kulkarni, R. Al-Rfou, B. Perozzi, and S. Skiena, "Statistically significant detection of linguistic change," in *Proceedings of the 24th International Conference on World Wide Web*, ser. WWW '15. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2015, pp. 625–635.

[17] A. Muromägi, K. Sirts, and S. Laur, "Linear ensembles of word embedding models," in *Proceedings of the 21st Nordic Conference on Computational Linguistics*. Gothenburg, Sweden: Association for Computational Linguistics, May 2017, pp. 96–104.

[18] M. Artetxe, G. Labaka, and E. Agirre, "Learning principled bilingual mappings of word embeddings while preserving monolingual invariance," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2289–2294.

[19] C. Xing, D. Wang, C. Liu, and Y. Lin, "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*. Denver, Colorado: Association for Computational Linguistics, May–Jun. 2015, pp. 1006–1011.

[20] P. H. Schönemann, "A generalized solution of the orthogonal procrustes problem," *Psychometrika*, vol. 31, no. 1, pp. 1–10, Mar 1966.

[21] M. D. Tredici and R. Fernández, "Semantic variation in online communities of practice," in *IWCS 2017 - 12th International Conference on Computational Semantics - Long papers*, 2017. [Online]. Available: https://www.aclweb.org/anthology/W17-6804

[22] A. Hätty, D. Schlechtweg, and S. Schulte im Walde, "SURel: A gold standard for incorporating meaning shifts into term extraction," in *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 1–8.

TABLE II
RANKED LIST OF TARGET WORDS BASED ON THEIR AMBIGUITY SCORES

Light controller		Mechanical CAD		Medical Software		Athletes Network	
Term	Score	Term	Score	Term	Score	Term	Score
express	0.7686	thread	1.0811	compression	1.0963	delivery	1.0899
net	0.763	kingdom	0.9953	mouse	1.0841	circuit	1.0893
grid	0.7314	united	0.8992	assembly	0.9898	stable	1.049
deal	0.7037	net	0.8177	native	0.9866	sun	1.0488
family	0.6934	background	0.7982	controlled	0.9639	single	1.0144
united	0.6925	uniform	0.7864	root	0.9551	state	1.0122
primary	0.6902	translation	0.771	internal	0.911	deep	0.9946
reverse	0.6855	natural	0.7427	spectrum	0.8867	drive	0.9844
universal	0.6677	record	0.7376	bank	0.8825	free	0.9687
life	0.6665	stand	0.7167	derivative	0.8823	active	0.9612
⋮		⋮		⋮		⋮	
typical	0.1355	notion	0.1569	michael	0.1601	occurs	0.2409
specialized	0.1286	leave	0.1523	government	0.1601	interview	0.2356
richard	0.1272	appeared	0.149	david	0.1568	daniel	0.2353
compatible	0.1257	david	0.1464	political	0.1559	measured	0.2277
detect	0.125	nature	0.1415	required	0.1539	located	0.2246
benefit	0.1223	requiring	0.1395	dependent	0.1497	prevent	0.2225
infrastructure	0.115	corresponds	0.1382	charles	0.1453	increase	0.1992
corresponding	0.1145	authority	0.1343	james	0.1417	federal	0.1885
david	0.112	required	0.1321	peter	0.1361	growth	0.1876
authority	0.109	corresponding	0.1176	robert	0.1271	robert	0.1866

Medical Device		Medical Robot		Sports Rehab Machine	
Term	Score	Term	Score	Term	Score
root	0.8802	stroke	0.919	kingdom	0.907
mouse	0.8633	kingdom	0.893	stroke	0.8495
kingdom	0.8383	vessel	0.8651	progressive	0.8414
iron	0.8381	thread	0.8385	net	0.8334
internal	0.8043	floating	0.8045	suspension	0.8322
progressive	0.7957	strain	0.8018	induction	0.8244
agent	0.7875	mouse	0.7997	thread	0.8236
express	0.7733	progressive	0.7983	root	0.8093
plasma	0.7685	die	0.787	transmission	0.7871
net	0.7678	secondary	0.786	die	0.7821
⋮		⋮		⋮	
argued	0.1631	corresponding	0.1695	corresponding	0.196
richard	0.1608	corresponds	0.1693	told	0.1958
authority	0.1606	feel	0.167	joseph	0.1956
michael	0.1569	coating	0.1666	understanding	0.1953
required	0.154	wife	0.1603	love	0.1902
peter	0.1388	michael	0.159	economic	0.1902
robert	0.1381	authority	0.156	coating	0.1892
david	0.1201	peter	0.1437	pay	0.1856
james	0.1188	david	0.1412	authority	0.1838
charles	0.1157	required	0.1382	causing	0.1822

TABLE III
CASES OF DISAGREEMENT BETWEEN THE LINEAR TRANSFORMATION APPROACH (R_1) AND THE ONE SUGGESTED BY FERRARI *et al.* (2018) (R_2)

Light Controller (986)				Mechanical CAD (1016)				Medical Software (742)			
Term	R_1	R_2	$\lvert R_1 - R_2 \rvert$	Term	R_1	R_2	$\lvert R_1 - R_2 \rvert$	Term	R_1	R_2	$\lvert R_1 - R_2 \rvert$
robert	972	31	941	notion	1007	18	989	peter	741	12	729
michael	933	4	929	third	31	1008	977	thomas	727	14	713
phenomenon	971	60	911	kingdom	2	975	973	third	18	730	712
peter	902	8	894	richard	964	11	953	richard	712	19	693
wide	45	939	894	green	40	987	947	mind	707	38	669

Athletes Network (569)				Medical Device (1168)				Medical Robot (1507)				Sports Rehab Machine (1624)			
Term	R_1	R_2	$\lvert R_1 - R_2 \rvert$	Term	R_1	R_2	$\lvert R_1 - R_2 \rvert$	Term	R_1	R_2	$\lvert R_1 - R_2 \rvert$	Term	R_1	R_2	$\lvert R_1 - R_2 \rvert$
daniel	562	1	561	peter	1164	23	1141	paul	1486	8	1478	daniel	1613	3	1610
robert	569	17	552	third	56	1162	1106	coating	1501	29	1472	love	1619	17	1602
main	28	537	509	richard	1160	76	1084	peter	1505	38	1467	peter	1591	16	1575
effect	522	15	507	white	74	1150	1076	third	42	1504	1462	coating	1621	66	1555
child	59	558	499	chess	1102	39	1063	richard	1482	22	1460	told	1616	73	1543