A Deep Context-wise Method for Coreference Detection in Natural Language Requirements

Yawen Wang[†‡], Lin Shi[†‡*], Mingyang Li[†‡], Qing Wang[†‡§*], Yun Yang[¶]
[†]Laboratory for Internet Software Technologies, Institute of Software, Chinese Academy of Sciences, Beijing, China
[‡]University of Chinese Academy of Sciences, Beijing, China
[§]State Key Laboratory of Computer Sciences, Institute of Software, Chinese Academy of Sciences, Beijing, China
[¶]School of Software and Electrical Engineering, Swinburne University of Technology, Australia
{yawen, mingyang}@itechs.iscas.ac.cn, {shilin, wq}@iscas.ac.cn, yyang@swin.edu.au

***Abstract*—Requirements are usually written by different stakeholders with diverse backgrounds and skills and evolve continuously. Therefore inconsistency caused by specialized jargons and different domains, is inevitable. In particular, entity coreference in Requirement Engineering (RE) is that different linguistic expressions refer to the same real-world entity. It leads to misconception about technical terminologies, and impacts the readability and understandability of requirements negatively. Manual detection entity coreference is labor-intensive and time-consuming. In this paper, we propose a DEEP context-wise semantic method named DEEPCOREF to entity COREFerence detection. It consists of one fine-tuning BERT model for context representation and a Word2Vec-based network for entity representation. We use a multi-layer perception in the end to fuse and make a trade-off between two representations for obtaining a better representation of entities. The input of the network is requirement contextual text and related entities, and the output is the predictive label to infer whether two entities are coreferent. The evaluation on industry data shows that our approach significantly outperforms three baselines with average precision and recall of 96.10% and 96.06% respectively. We also compare DEEPCOREF with three variants to demonstrate the performance enhancement from different components.**

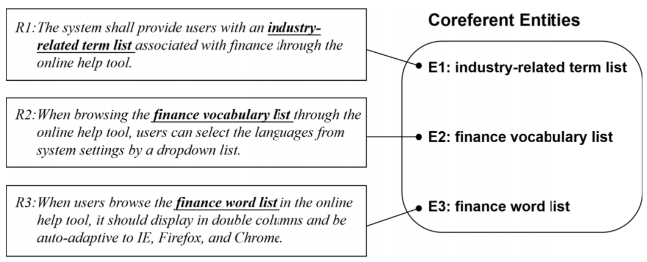
***Index Terms*—Requirement engineering, entity coreference, deep learning, fine-tuning BERT**

I. INTRODUCTION

Most software requirements are specified in natural language with the flexibility to accommodate the arbitrary abstraction [1]–[4]. It is a challenging but essential task to write requirements clearly without inconsistency and ambiguity before passing to the later stages of the development [5], [6]. The inconsistency, which is one of the quality principles related to linguistic aspects of natural language requirements [7], might occur between requirements analysts and domain experts because of their specialized jargons, or stakeholders from different domains.

In particular, stakeholders could use different linguistic expressions to refer to the same real-world entity in natural language requirements, and we define such phenomena as Entity Coreference (EC). More specifically, Figure 1 presents an example of EC. The three requirements have their related entities: "industry-related term list" in R1, "finance vocabulary

[*] Corresponding authors.



*R1: The system shall provide users with an **industry-related term list** associated with finance through the online help tool.*

*R2: When browsing the **finance vocabulary list** through the online help tool, users can select the languages from system settings by a dropdown list.*

*R3: When users browse the **finance word list** in the online help tool, it should display in double columns and be auto-adaptive to IE, Firefox, and Chrome.*

Fig. 1. Examples of coreferent entities in textual requirements, which make the requirements difficult to understand.

list" in R2 and "finance word list" in R3. However, according to their contexts, the three entities refer to the same thing. EC might lead to misconception on entities, thus impairing the readability and understandability of requirements. This work takes the first step to resolve EC in RE.

In the literature, some works have been proposed to tackle the problem of inconsistency or ambiguity in textual requirements. Pattern-based methods [8]–[15] use Part-of-Speech (POS) patterns or heuristics. Learning-based methods [5], [16], [17] use information retrieval (IR) technique such as Latent Semantic Indexing (LSI) or unsupervised clustering algorithms. Similarity-based methods include word embeddings [3] and syntactic methods (e.g., Jaccard [18] and Levenstein [19]). However, these methods cannot be directly utilized in EC due to the following challenges:

- **Multi-word entity.** In textual requirements, entities are more about noun phrases [20], [21] than a single word. As shown in Figure 1, all entities in examples consist of multiple words. Based on observations of our industry data, the average length of entities is 3.52. Multi-word entities are difficult to represent with word-level representation. For example, although E1 refers to the same entity as E2 and E3, E1 is quite different from the other two expressions that they only share one identical word "list". If we simply use the word-wise similarity methods such as word embedding, incorrect EC will be given that E2 and E3 are coreferent while E1 is a different entity.
- **Missing contextual semantics.** Existing solutions lack sentence-level contextual semantic information, which

can provide extra information for resolving EC. In most cases, we infer whether two entities are coreferent based on their contexts. Coreferent entities usually have similar contexts. For example, all the three requirements in Figure 1 have similar contextual words such as "user", "online help tool", etc., which indicate three entities are coreferent. Therefore, how to fuse contextual semantic in entity representation is important as well.

- **Insufficient annotated resources.** EC detection in RE is a domain-specific task, which cannot directly benefit from large general corpora or public knowledge bases like general coreference detection tasks. In addition, annotating coreferent entities in requirements requires domain expertise and intensive manual effort, resulting in insufficient annotated data for effective learning. How to use limited annotation data and benefit from pre-trained models trained on large general corpora is difficult.

We propose a DEEP context-wise semantic method named DEEPCOREF to resolve entity COREFerence in natural language requirements. First, we truncate context for each entity and then convert $\langle context, entity\rangle$ pairs to model input format. Then we build a context-wise similarity network to infer whether two entities are coreferent. The network consists of two parts. One is a deep fine-tuning BERT context model for context representation, and the other is a Word2Vec-based entity network for entity representation. Finally, we use a Multi-Layer Perceptron (MLP) to fuse two representations. The input of the network is requirement contextual text and related entities, and the output is the predictive label to infer whether two entities are coreferent. The combined sentence-level context representation and word-level entity representation can be trained jointly with other parameters, thus obtaining a better entity representation. In addition, with models pre-trained on large corpora (e.g., BERT and word embeddings) and fine-tuning technique, we only need to annotate small amounts of data for fine-tuning. It alleviates insufficient annotated resource problems and the high cost of manual annotation as well.

We investigate the effectiveness of DEEPCOREF with data from our industry partner. The experimental results show that our approach significantly outperforms three baselines with average precision and recall of 96.10% and 96.06%. The results confirm that our approach could effectively detect EC from textual requirements, thus can facilitate reaching a shared understanding on entities among multiple stakeholders from different domains in an automated way. We also compare DEEPCOREF with three variants to demonstrate the performance enhancement from different components.

The main contributions of this paper are as follows:

- We highlight the importance of detecting EC in RE.
- A deep context-wise semantic method with a powerful representation for entities in textual requirements for automatically detect EC.
- Experimental evaluation on 21 projects with 1853 samples from the industry community with promising results.
- Public-access of source code[1] to facilitate the replication of our study and its application in other contexts.

The rest of the paper is organized as follows. Section II describes the background. Section III presents the design of our proposed approach. Sections IV and V show the experimental setup and evaluation results respectively. Section VI provides a detailed discussion. Section VII describes threats to validity. Section VIII surveys related work. Finally, we summarize the paper in Section IX.

II. BACKGROUND

This section describes key techniques related to this research: word embeddings, fine-tuning BERT and Coreference Resolution (CR). We include them here because our work is based on these techniques.

A. Word Embeddings

Embedding (also known as distributed representation [22], [23]) is a technique for learning vector representations of entities such as words, sentences and images in such a way that similar entities have vectors close to each other [22], [24]. A typical embedding technique is word embedding, which represents words as fixed-length vectors so that similar words are close to each other in the vector space [22], [24], [25]. Comparing with Levenstein [19], here "similar" means semantic similarity instead of string similarity. Word embeddings are based on the distributional hypothesis of Harris [26]. We can estimate distances and identify semantic relations from their vectors.

Word embedding is usually implemented by a model such as Continuous Bag-of-Words (CBOW) and Skip-Gram [24]. These models build a neural network that captures the relations between a word and its contextual words. The vector representations of words, as parameters of the network, are trained with a text corpus [22]. Another word embedding model is GloVe [25], which is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

Information captured from corpora substantially increases the value of word embeddings to both unsupervised and semi-supervised Natural Language Processing (NLP) tasks. For example, good representations of both the target word and the given context are helpful to various tasks, including word sense disambiguation [27], coreference resolution and named entity recognition (NER) [23], [28], [29]. The context representations used in such tasks are commonly just a simple collection of the individual embeddings of the neighboring words in a window around the target word, or a (sometimes weighted) average of these embeddings [30]. Likewise, a sentence (i.e., a sequence of words) can also be embedded as a vector [31]. A simple way of sentence embedding is, for example, to consider it as a bag of words and add up all its word vectors [32].

[1]https://github.com/MeloFancy/DeepCoref

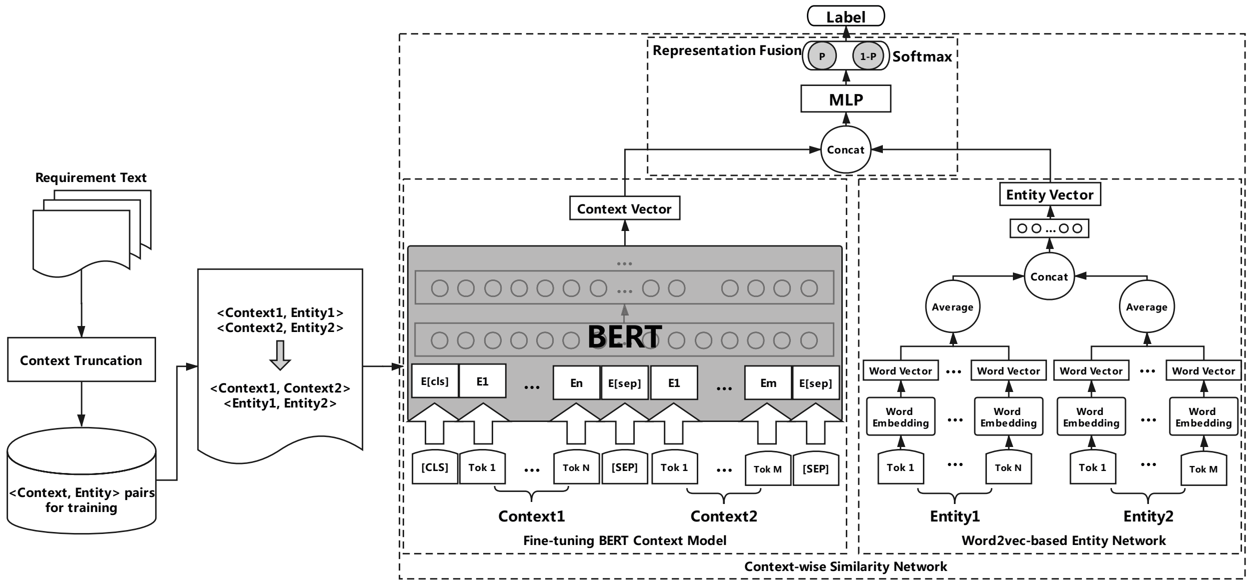


Fig. 2. The Overview of DEEPCOREF

B. Fine-tuning BERT

BERT (Bidirectional Encoder Representations from Transformers) [33] is a deep bidirectional transformer encoder [34] trained with the objective of masked language modeling and the next-sentence prediction task, which proves effective in various NLP tasks.

BERT framework has two steps: 1) pre-training, where the model is trained on unlabeled data over different pre-training tasks. 2) fine-tuning, where the BERT model is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream tasks. BERT has two model sizes: $BERT_{BASE}$ (L=12, H=768, A=12, Total Parameters=110M) and $BERT_{LARGE}$ (L=24, H=1024, A=16, Total Parameters=340M), where the number of layers (i.e., Transformer blocks) is denoted as L, the hidden size as H, and the number of self-attention heads as A.

BERT is designed to unambiguously represent both a single sentence and a pair of sentences in one token sequence, for handling a variety of downstream tasks. As for output, the token representations are fed into an output layer for token-level tasks, and the $[CLS]$ representation is fed into an output layer for classification. The pre-trained BERT can be simply plugged by the task-specific inputs and outputs and fine-tuned all the parameters end-to-end, which is relatively inexpensive compared to pre-training.

C. Preliminaries on Coreference Resolution

Coreference is defined as occurring when one or more expressions in a document refer to one entity. CR is a classical NLP task of finding all expressions that are coreferent with any of the entities found in a given text [35]–[38]. In CR, an entity refers to an object or set of objects in the world, while a mention is the textual reference to an entity [36].

There two types of tasks in CR [37]: 1) resolving entities versus events 2) whether co-referring mentions occur within a single document (WD: within-document) or across a document collection (CD: cross-document). Compared to entity CR, event coreference is considered to be a more difficult task, mostly due to the more complex structure of event mentions [37], [39]. Entity mentions are mostly noun phrases, while event mentions may consist of a verbal predicate (acquire) or a nominalization (acquisition), where these are attached to arguments, including event participants and spatio-temporal information [37]. WDCR approaches provide techniques for the identification of mentions in one document that refer to the same underlying entity/event, while CDCR approaches provide techniques for the identification of mentions in different documents [38].

III. APPROACH

To address the challenges mentioned in Section I, we propose an approach named DEEPCOREF for resolving EC detection. Figure 2 presents the overview of DEEPCOREF. Given a set of textual requirements written in natural language and its related entity, we firstly truncate their corresponding contexts (see Section III-A). Then we build a context-wise similarity classification network (see Section III-B) to predict whether a pair of entities are semantically equivalent. The network mainly consists of two parts. One is a deep fine-tuning BERT model for encoding the contexts, the other is a Word2Vec-based network for encoding entities. The output of two parts is representations of contexts and entities respectively, which are then fed into an MLP for similarity classification. Finally, we infer the predictive class based on the probabilities produced by the softmax layer.

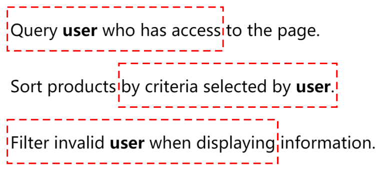

Query **user** who has access to the page.
Sort products by criteria selected by **user**.
Filter invalid **user** when displaying information.

Fig. 3. An example of context truncation. The bold word (e.g., user) is entity word. The red dotted rectangle represents the window. Here window size $M = 5$, and the length of entity $N = 1$.

A. Context Truncation

Since entity extraction has been widely developed by many NLP researches [20], [21], [40], DeepCoref does not focus on entity extraction, and utilizes entities that have already been extracted as the basic data. In our study, entities are ready-made and manually reviewed to avoid error accumulation from entity extraction tools.

In this study, the context refers to the neighboring words in a window around a certain entity. This step is to truncate requirement text centered on an entity with a window size as the context related to the entity. Given an entity and its related requirement text, we first locate the entity and then truncate text centered on the entity according to the window size. Entities might occur in different positions of one sentence (e.g., near the beginning, near the middle and near the end). So we tackle different cases according to the rules below. We assume window size is M, the length of entity denoted as N, the length of text sequence before entity denoted as l_{pre}, the length of text sequence after entity denoted as l_{sub}:

- If $l_{pre} \geqslant \lceil \frac{M-N}{2} \rceil$ and $l_{sub} \geqslant \lceil \frac{M-N}{2} \rceil$, both previous and subsequent text sequences are truncated by length $\lceil \frac{M-N}{2} \rceil$.
- If $l_{pre} \geqslant \lceil \frac{M-N}{2} \rceil$ and $l_{sub} < \lceil \frac{M-N}{2} \rceil$, the previous text sequence is truncated by length $\min(l_{pre}, M-N-l_{sub})$, and all subsequent words are reserved, where $min(\cdot)$ is to take the minimum.
- If $l_{pre} < \lceil \frac{M-N}{2} \rceil$, the previous text sequence is truncated by length l_{pre}, and all subsequent words are reserved.

The final extracted context is a concatenation of truncated previous sequence (denoted as pre), the entity itself and truncated subsequent sequence (denoted as sub): $[pre \oplus entity \oplus sub]$. Finally, we use a special symbol $[PAD]$ padding to the length of window size. Figure 3 demonstrates an example of context extraction for each case. By context truncation, we obtain the entity and its related context (e.g., $\langle context, entity \rangle$).

B. Build Context-wise Similarity Network

The context-wise similarity network takes two pairs (e.g., $\langle context_1, entity_1 \rangle$ and $\langle context_2, entity_2 \rangle$) as input and predicts whether two pairs are coreferent. The network consists of two parts. One is a fine-tuning BERT model for learning context representations, and another is a Word2Vec-based network for learning entity representations. We concatenate two representations for better combining semantic information about the entire contextual sentences and individual words. Finally, we use an MLP and softmax layer to infer the predictive label.

1) Fine-tuning BERT Context Model: A powerful context representation is helpful for measuring context-wise similarity [41]. In many NLP tasks (e.g., entity disambiguation and entity coreference resolution), the context representations are commonly a collection of the individual embedding of contextual words, (e.g., a weighted average of these embeddings). Such approaches do not include any mechanism for optimizing the representation of the entire contextual sentences [30].

To obtain a good context representation, we use BERT which is a fine-tuning based and bidirectional pre-training representation model [33]. It takes a sentence pair (e.g., $\langle context_1, context_2 \rangle$, discomposed from two $\langle context, entity \rangle$ pairs) as input, and produces a context vector representation. Due to limited computing resources, we use the model $BERT_{BASE}$ with a relatively small model size, which has 12 layers, hidden dimension size 768 and 12 attention heads. In BERT, the input can be a pair of sentences. Each sentence is represented by 128 word-piece tokens (window size $M = 128$ in Section III-A). Two contexts are concatenated and fed to the model as a sequence pair together with special start and separator tokens: $([CLS]\ context_1\ [SEP]\ context_2\ [SEP])$. The transformer encoder produces a context vector representation (denoted as v_{ctx}) of the input pair, which is the output of the last hidden layer at the special pooling token $[CLS]$ [33], [42].

2) Word2Vec-based Entity Network: To capture the word-level information of entities, we also build a Word2Vec-based network to learn an entity representation using word embeddings [22]. It takes an entity pair (e.g., $\langle entity_1, entity_2 \rangle$, discomposed from two $\langle context, entity \rangle$ pairs) as input, and produces an entity vector representation. We use the 300-dimensional word embeddings which are pre-trained on a 1.3G Wikipedia corpus[2] with 223M tokens and 2129K vocabularies. It is trained with three features (word features, n-gram features and character features) using the skip-gram model with negative sampling [43].

For each entity in the pair $\langle entity_1, entity_2 \rangle$, we first segment words and obtain the word embedding of each word. Then we use the average of embeddings of all words in one entity to represent the embedding of this entity (denoted as te). So the entity pair can be represented as $pe = [te_1 \oplus te_2]$. Because the dimension of word embeddings is 300, the dimension of te is 300 and the dimension of pe is 600. After that, pe is fed into a fully connected layer to produce an entity vector representation (denoted as v_t).

3) Representation Fusion: The output of two parts of context-wise similarity network: v_{ctx} is a representation of context pair, and v_t is a representation of entity pair. We need to fuse two representations to obtain semantic information in both sentence level and word level. The output is the label which represents whether two entities are coreferent.

[2]https://dumps.wikimedia.org/

First, we concatenate v_{ctx} and v_t ($v_f = [v_{ctx} \oplus v_t]$). Then we input v_f into MLP. MLP has three layers:

- **A fully connected layer**, which is to fuse v_{ctx} and v_t into one vector by $w^\top v_f$, where w is a learned parameter vector. w can be trained to make a trade-off between v_{ctx} and v_t.
- **A dropout layer**, which is used to avoid over-fitting [44] by randomly masking some neuron cells.
- **An output layer**, which transforms the vector into a 2-dimensional vector $[s_1, s_2]$, representing two labels (coreferent or non-coreferent).

The output of MLP is a similarity measure $[s_1, s_2]$ that represents the scores of the two classes respectively, where $s_i \in R$. Finally, we perform softmax on this 2-dimensional vector, which can be specified as:

$$Softmax(s_i) = \frac{e^{s_i}}{\sum_{j=1}^{2} e^{s_j}}$$

Then $[s_1, s_2]$ can be normalized to probabilities $[p, 1-p]$, where $p \in [0, 1]$.

C. Implementation

We implement our approach DeepCoref using Transformers[3] which is an open-source library for Natural Language Understanding (NLU) and Natural Language Generation (NLG) with over 32+ pre-trained models built on Pytorch[4].

Training Details: As for some crucial settings, the learning rate is set 10^{-5}. The optimizer is Adam [45] algorithm. We use the mini-batch technique for speeding up the training process with batch size 8. The drop rate is 0.1, which means 10% of neuron cells will be randomly masked to avoid over-fitting. Since the task is a classification problem, we use cross-entropy as the loss function, which is specified as:

$$Loss = \sum_x p(x) \cdot log(\frac{1}{q(x)})$$

where $p(x)$ and $q(x)$ are the probability distribution of predicted label and ground-truth label respectively.

The design of the context-wise similarity network makes all parameters jointly fine-tuned on a specific task (e.g., similarity classification), which can benefit from large corpora pre-training in a relatively inexpensive way. It also alleviates insufficient annotated resource problems to some extent. Parameters in BERT are fine-tuned to obtain a better context representation according with specific tasks and data. Parameters in Word2Vec-based network are trained to obtain a better entity representation based on pre-trained word embeddings. Parameters in MLP are trained to better fuse both representations, and make a trade-off between two representations to reach a more accurate classification result.

[3]https://github.com/huggingface/transformers

[4]https://pytorch.org

IV. Experiment Design

A. Research Questions

Our evaluation addresses the following three research questions.

- **RQ1 (Advantage)** Can DeepCoref outperform existing techniques on coreference detection?

To investigate the advantage of our approach, we conduct 10-fold cross-validation on EC detection using data from our industry partner. We compare the performances of three baselines (see Section IV-C). These approaches include syntactic or semantic similarity measures to detect coreference on word level or sentence level. We compare these approaches to demonstrate the advantage of combining context and entity representations. Besides, we also present statistical results by the project to examine the stability and generalizability across different projects.

- **RQ2 (Effectiveness)** How effective does each component facilitate EC detection?

To examine the performance enhancement introduced by context representations and entity representations respectively, we construct three variants: **DeepCoref-ctx**, which only contains the fine-tuning BERT model for context representations without the Word2Vec-based network for entity representations. **DeepCoref-entity**, which is totally opposite to DeepCoref-ctx only with Word2Vec-based network but BERT model (see Section IV-D). In addition, to demonstrate the advantage of fine-tuning BERT over IR-based technique for context representations, we build **DeepCoref-LSI** by repacing the BERT with LSI to produce context vectors. We conduct 10-fold cross-validation on DeepCoref, DeepCoref-ctx, DeepCoref-entity, and DeepCoref-LSI respectively to demonstrate the effectiveness for combining two representations.

- **RQ3 (Sensitivity)** To what degree does data size influence experimental results?

In RQ3, we conduct an experiment by increasingly enlarging the size of data to examine the sensitivity between performance enhancement and data augmentation. The amount of data verifies whether fine-tuning approaches can alleviate low-resource problem.

B. Data Preparation

Our experimental data is collected from the repositories of China Merchants Bank (CMB)[5]. We retrieve 21 projects from its repository, and each project has a set of requirement texts with corresponding entities that occurred in that text. The total number of text-entity pairs is 1949. The entities are recognized by requirement engineers, audited by project management department and well-maintained through the requirement evolution. We prepare data in the following steps:

[5]It is one of the world's top five hundred commercial banks.

1) Pre-processing: For each entity and its related requirement text, we filter noisy tokens such as URL, HTML tags and SQL statements using regular expressions. These tokens are produced by their management system but not removed when dumped from the system. Then we filter template words (e.g., "I would like to") which cannot contribute to the result but introduce noise, especially for contextual semantic similarity.

2) Sampling: After pre-processing, we truncate the context for each entity, which is demonstrated in Section III-A. We obtain 1949 $\langle context, entity\rangle$ (denoted as *ctx-entity* pair) pairs in total across all projects. Entities from different projects are definitely different no matter how similar their semantics are, so we sample two *ctx-entity* (e.g., $\langle$*ctx-entity*$_1$, *ctx-entity*$_2\rangle$) from the same project in a combination way. The combination step is to build relations among $\langle context, entity\rangle$ pairs for assigning labels.

3) Ground-truth Labeling: These requirements and entities are domain-specific, so it is challenging for data labeling. To guarantee the accuracy of the labeling results, the labeling process follows two steps: 1) The project management department of CMB assigns samples (e.g., $\langle$*ctx-entity*$_1$, *ctx-entity*$_2\rangle$ pairs) as well as original requirement texts for reference to requirement engineers according to the project team so that each annotator can label the samples belonging to his/her own products. 2) The labeling results withdrawn from each project team are reviewed by the project management department. Only those samples where both teams make a full agreement can be included in our dataset. As for samples which are annotated to different labels, two teams would discuss and decide through voting.

The two classes are extremely imbalanced (e.g., the majority of the samples are non-coreferent) after labeling. We use under-sampling technique to balance two classes. Finally, we obtain 1853 labeled samples ($\langle$*ctx-entity*$_1$, *ctx-entity*$_2$, *label*$\rangle$). The positive labels (897, 48.41%) mean *ctx-entity*$_1$ and *ctx-entity*$_2$ are coreferent, negative labels (956, 51.59%) for non-coreference.

C. Baselines

To further demonstrate the advantages of DEEPCOREF, we compare it with three commonly-used techniques for coreference detection.

Word2Vec: Ferrari et al. [3] present an approach based on word embeddings to support the identification of potentially ambiguous terms in the context of requirements elicitation interviews and group meetings. The "terms" in their context is still word-level, or the word itself is a phrase processed by word segment or text chunking. Word embedding provides a good semantic representation on word level. However, in our work, entities are not just single words but several words. We use an average of word embeddings to represent an entity, and then compute a similarity score for coreference detection.

LSI: Falessi et al. [17] use LSI to identify equivalent requirements after comparing several NLP techniques on a given dataset. It is an IR-based semantic sentence-level approach for representing a set of documents as vectors in a common vector space. LSI has been employed in a wide range of software engineering activities such as categorizing source code files [46], detecting high-level conceptual code clones [47], and recovering traceability links between documentation and source code [48], which is considered to be able to resolve the polysemy problem as well [49]–[51]. We build an LSI model to demonstrate its capability for context representations.

Levenstein: It is a syntactic similarity measure by calculating a score for a given pair of entities by finding the best sequence of edit operations to convert one entity into the other [19], [20]. We use the implementation in library Distance[6].

D. Experimental Setup

We conduct 10-fold cross-validation [52] on the dataset collected from CMB in RQ1 and RQ2. We randomly divide our dataset into ten parts. We use nine of those parts for training and reserve one part for testing. We repeat this procedure 10 times each time reserving a different part for testing. All the experiments are conducted based on the same data folds to avoid the impact of different data partitions. In RQ3, we randomly split data into the training set and testing set according to a training set ratio, which indicates that we use the ratio percentage of data to train model and evaluate the remaining testing set. The experimental environment is a desktop computer equipped with a NVIDIA 1060 GPU, intel core i7 CPU, 16GB RAM, running on Ubuntu OS.

Experiment I (Advantage): To demonstrate the advantage of DEEPCOREF, we compare the performance of DEEPCOREF with three approaches (Word2Vec, LSI, Levenstein distance). Word2Vec uses an average of word embeddings to represent an entity which is to investigate the performance of entity representation with word embeddings. LSI is built on the concatenation of both context and entity to investigate the performance of the combination of context and entity representation with LSI. Levenstein is used to measure the distance between entities to investigate the performance of simple syntactic approaches. The output of Levenstein is a similarity score. The outputs of Word2Vec and LSI are vector representations, and subsequently, we infer the predicted label by computing similarity score via cosine similarity [53]. So we need a similarity ratio to decide whether two entities are coreferent, which should be tuned carefully. We set the ratio of Word2Vec, LSI and Levenstein as 0.85, 0.67 and 0.25 respectively. We investigate the ratio selection in Section VI-A. In addition, we give statistical results by the project to examine the stability and generalizability across different projects.

Experiment II (Effectiveness): We demonstrate the performance enhancement introduced by each component by constructing DEEPCOREF-ctx and DEEPCOREF-entity. DEEPCOREF-ctx only keeps BERT model and DEEPCOREF-entity only keeps Word2Vec network. In addition, we build DEEPCOREF-LSI, which is a variant of DEEPCOREF by replacing the BERT with LSI, and other parts remain unchanged. DEEPCOREF-LSI is to demonstrate the performance enhancement from BERT for computing context representation.

[6]https://github.com/doukremt/distance

Experiment III (Sensitivity): We conduct an experiment by increasingly enlarging the size of data to examine the sensitivity between performance enhancement and data augmentation. We present the time consumption of each experiment as well. The experiment is conducted by splitting all data into two parts (training set and testing set) randomly according to the training set ratio which is increased from 5% to 90%. For each ratio, we perform five experiments with a boxplot to show the evaluation metrics, and take the average time of five experiments as consuming time.

E. Evaluation Metrics

We use precision recall, and F1-Score, which are commonly-used metrics, to evaluate the performance of DEEPCOREF. We mentioned that we collect and annotate data in cooperation with CMB (see Section IV-B). Given the ground-truth label and predicted label from DEEPCOREF, we compute the metrics of all testing data for each round of 10-fold cross-validation to measure the performance. As for the performance by the project in RQ1, we compute the metrics for each project.

1) Precision, which refers to the ratio of the number of correct predictions of positive labels to the total number of predictions of positive labels. 2) Recall, which refers to the ratio of the number of correct predictions of positive labels to the total number of positive labels. 3) F1-Score, which is the harmonic mean of precision and recall.

We calculate metrics for each label and take their unweighted mean as final results. In addition, some baseline approaches need to measure the similarity to decide whether two entities are coreferent, so we use cosine similarity [53] to compute the distance between two vector representations.

V. RESULTS AND ANALYSIS

A. Answering RQ1: Advantage of DEEPCOREF

Figure 4 presents the EC detection performance on DEEPCOREF and baselines respectively across the 10-fold cross-validation. We can see that DEEPCOREF can achieve 96.10% precision and 96.06% recall on average, which are much higher than other baselines. The precision and recall of Word2Vec are 84.57% and 84.21% respectively, LSI 84.12% and 84.01%, Levenstein 84.65% and 83.46%. In addition, the length of the box of DEEPCOREF is relatively lower than baselines, further signifying the stability of the performance.

Figure 5 presents the precision and recall by 21 projects. We can see both precision and recall of DEEPCOREF are more stable and higher than other baselines across projects. The text presentation styles are distinct in different projects, so the results of Word2Vec and Levenstein indicate large differences in performance on different projects. These two approaches lack sentence-level information of context, thus cannot capture the contextual semantic differences across projects only using entity information. LSI fluctuates largely in several projects although it can capture the sentential context semantics. This is mainly because LSI is constructed based on statistical information on current training data, the representation ability

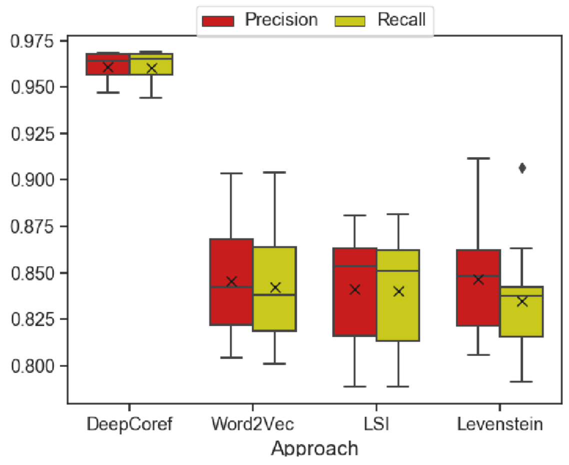


Fig. 4. RQ1: The advantage of DEEPCOREF over baselines. The cross is the mean value of 10-fold cross validation.

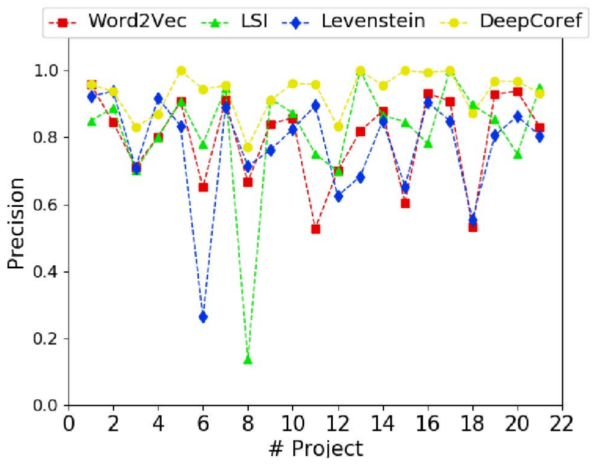
is less powerful than models pre-trained on large corpora and fine-tuned with training data. By contrast, DEEPCOREF which is more stable, obtains a more powerful representation by combining the context semantics, thus more adaptable to different presentation styles.

The reasons why DEEPCOREF noticeably outperforms the three baselines are: 1) DEEPCOREF uses both sentence-level and word-level semantics thus can capture more information from contexts and entities. 2) DEEPCOREF uses pre-trained models thus can benefit from large general corpora pre-training. 3) DEEPCOREF uses the fine-tuning technique, which can improve adaptation on domain-specific tasks.

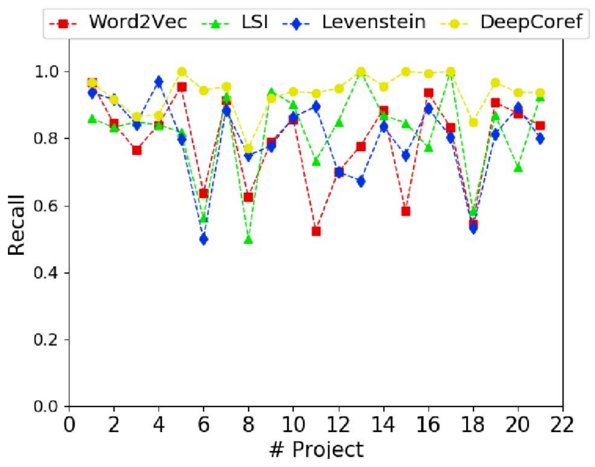
B. Answering RQ2: Effectiveness of DEEPCOREF

Figure 6 presents the performance on DEEPCOREF and three variants respectively across the 10-fold cross-validation. The average of precision and recall of DEEPCOREF-ctx reach 79.83% and 68.21%, DEEPCOREF-entity 63.17% and 61.77%, DEEPCOREF-LSI 66.25% and 62.62% respectively. The performance of DEEPCOREF is much higher and more stable than three variants.

The comparison among DEEPCOREF, DEEPCOREF-ctx and DEEPCOREF-entity indicates the performance enhancement from different components. More specifically, the fine-tuning BERT model improves the performance of precision and recall by 32.93% and 34.29% (differences between DEEPCOREF and DEEPCOREF-entity). The Word2Vec-based network improves performance by 16.27% and 27.85% (differences between DEEPCOREF and DEEPCOREF-ctx). The comparison between DEEPCOREF-ctx and DEEPCOREF-entity indicates that context semantics are more effective than entity semantics. The improvement of precision and recall reaches 16.66% and 6.44% respectively. The comparison between DEEPCOREF and DEEPCOREF-LSI indicates the stronger contextual representation from BERT than LSI, where the improvement reaches 29.85% and 33.44% respectively.



(a) Precision



(b) Recall

Fig. 5. RQ1: The advantage of DEEPCOREF over baselines by project. The number of projects is 21.

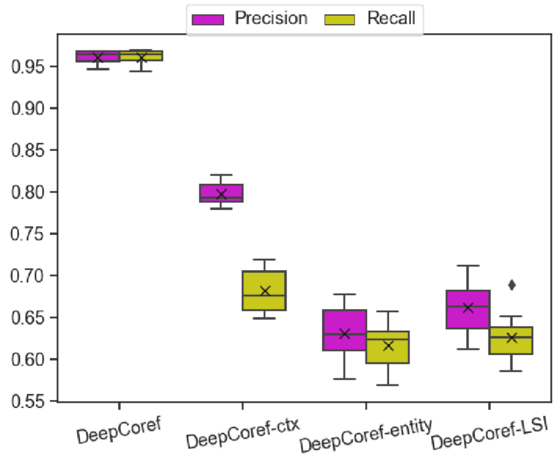


Fig. 6. RQ2: The performance of DEEPCOREF and its variants. The cross is the mean value of 10-fold cross validation.

In summary, each component of our network improves the performance to varying degrees. The combination can obtain a quite promising performance. The application of fine-tuning BERT model significantly enhances performance.

C. *Answering RQ3: Sensitivity of* DEEPCOREF

Figure 7 represents the relationship between performance and time consumption of DEEPCOREF when enlarging the size of the training set. When the training set ratio increases from 5% to 90%, the performance of DEEPCOREF rises sharply before 20% and has a small increase later. The variance of data at each point after 40% is also similar. The last point is the results coming from RQ1 for comparison, which shows the best performance. The results indicate that DEEPCOREF is not very sensitive after the training data is greater than 60% (i.e., around 1100 in our experimental settings). Moreover, we can find that just using 20% data to train, the performance is also greater than 92% on average. It demonstrates that DEEPCOREF can address the low-resource problem well. In addition, the

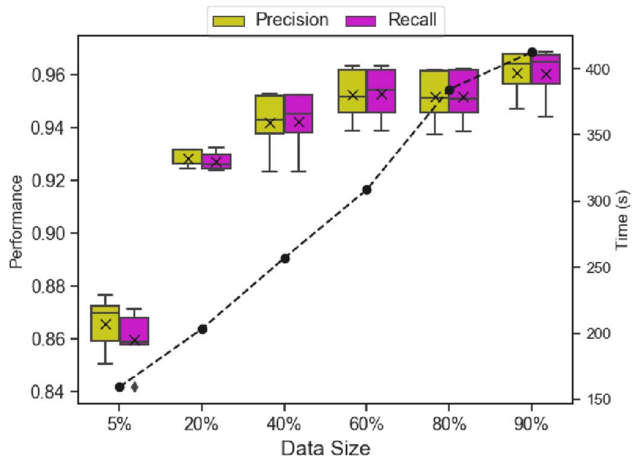


Fig. 7. RQ3: The performance of DEEPCOREF by data augmentation. The dotted line is time consuming.

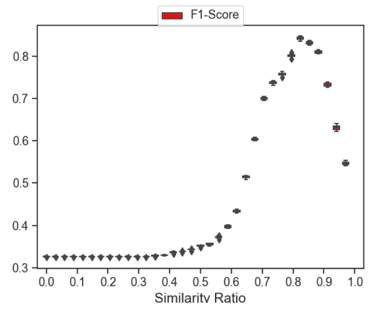
time consumption increases approximatively linearly from 159.42s to 412.52s.

In summary, benefiting from large corpora pre-training and the fine-tuning technique, DEEPCOREF can reach a promising performance on a relatively small dataset.

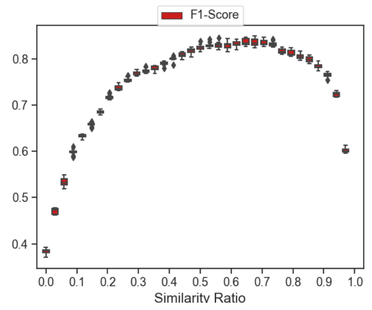
VI. DISCUSSION

A. *Parameter Settings on Baselines*

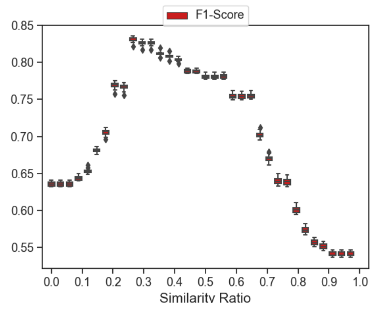
The performances of baselines are affected by the value selection of similarity ratios. Here we discuss the parameter determination process in our experiments. To achieve the best performance of these baselines, we conduct a set of experiments to find the sweet parameters. The best parameter settings are used in the comparison. Baselines are similarity-based methods, which are sensitive to the value of the ratio, so the parameter we analyze is similarity ratio. We vary the values of similarity ratios for Word2Vec, LSI, and Levenstein respectively, and evaluate their impact on the performance.



(a) Word2Vec



(b) LSI



(c) Levenstein

Fig. 8. The boxplot changing curve of F1-Score with the similarity ratio increasing from 0 to 1 by step 0.03 for each approach. Each box contains results of one 10-fold cross validation.

We present the box-plot changing curve (each box includes 10 results from 10-fold cross-validation) of F1-Score for each approach, when the ratio increases in $[0, 1]$ by step 0.01 (for readability, the step in the figure is 0.03). We also present the optimal value of the ratio for each round of 10-fold cross-validation of each approach. The final similarity ratio of each approach is computed by an average of 10 optimal values.

Figure 8 shows the F1-Score when the similarity ratio increases from 0 to 1 for each baseline. We can see that the ratio can influence the performance of these similarity-based approaches significantly, and the optimal values are distinct for each approach. Generally with the increase of similarity ratio, the F1-Score first rises and then declines for all approaches. Nevertheless, this general trend exhibits a slight difference among these approaches. For Word2Vec, the curve is steep, rising when the ratio is less than 0.85 and declining after that, which means that the optimal values of the ratio are stable around 0.85 for each round of 10-fold cross-validation. For LSI, the curve rises slowly before 0.5, then keeps steady between 0.5 and 0.7, and finally declines after 0.7. This means that the optimal values fluctuate in the interval $(0.5, 0.7)$ for all rounds. For Levenstein, the cure rises dramatically before 0.25, then declines slowly between 0.25 and 0.65, and declines dramatically after 0.65. The optimal values are around 0.25.

For each round of 10-fold cross-validation, the optimal similarity ratios of Word2Vec and Levenstein are the same values of 0.85 and 0.25 respectively, while the optimal ratio of LSI fluctuates slightly around 0.67, which is consistent with changing curves in Figure 8. The final ratio is computed by the average of these optimal values, where the ratios of Word2Vec, LSI and Levenstein are 0.85, 0.67 and 0.25. Hence one should carefully tune the similarity ratios for each approach, in order to achieve the best performance for a fair comparison.

B. Applicability

We list some key points when applying our method: 1) Our method is evaluated on short texts, where contexts can contain enough semantic information. When applying to long texts, some contexts truncated by window may lack useful information which is far from entities. Tuning window size might alleviate the problem. 2) Our data is from financial domain. One should annotate about 1000 samples for fine-tuning the whole model to tackle domain adaption. 3) The entities in our data are ready-made. If somenone wants to apply our method but has no entities, he/she needs to extract entities firstly using mature NLP techniques such as text trunking [20], [21] and POS patterns [40]. However, the error brought by these tools needs manual correction inevitably. 4) When applying to other languages, BERT and word embeddings must be pre-trained on corpus of corresponding language.

VII. Threats to Validity

External Validity: The external threats are related to the generalizability of the approach. Although the data is collected from the industry community, we retrieve as many projects as possible. The evaluation results by projects show that our approach is generalizable across projects, which largely alleviates the threat. In addition, our approach uses models pre-trained on large general corpus and the fine-tuning technique, which alleviates the low-resource and generalizability problem.

Internal Validity: The internal threats relate to experimental errors and biases. Threats to internal validity may come from the entity generation. The entities in our data are ready-made and well-maintained by our industry partners, which has a slight impact on our results.

Construct Validity: The construct threats relate to the suitability of evaluation metrics. We utilize precision and recall for evaluation, where we use cosine similarity to measure whether two entities are coreferent. The threats might come from the selection of similarity ratio. To reduce that threat, we perform an experiment on tuning ratios and use the average of optimal values as ratios (see Section VI-A). In addition, both predictive positive and negative labels are equally important in predictions, so we calculate the evaluation metrics for each label and take their unweighted mean as final results.

VIII. Related Work

Our work is related to previous studies that focused on 1) detection of inconsistency in requirements written in natural language; and 2) coreference resolution. We briefly review the recent works in each category.

A. Detection of Inconsistency

The amount of research on inconsistency detection has increased significantly in the past years. Mezghani et al. [5] used unsupervised machine learning algorithm, k-means, for a redundancy and inconsistency detection in the RE context. They introduced a filtering approach to eliminate "noisy" requirements and a pre-processing step based on the NLP technique and used POS tagging and noun chunking to detect technical business terms. PBURC [16] is a pattern-based unsupervised requirements clustering framework (based on k-means algorithm), which makes use of machine-learning methods for requirements validation. The approach aimed to overcome data inconsistencies and effectively determine appropriate requirements clusters for the optimal definition of software development sprints. Traditional techniques such as bag-of-words (BOW), Term Frequency and Inverse Document Frequency (TF-IDF) frequency matrix and n-gram language modeling were firstly used on redundancy detection. Juergens et al. [54] found that clone detection, a technique widely applied to source code, is promising to assess redundancy in an automated way. They used ConQAT to identify copy&paste operations in software requirements specifications. Falessi et al. [17] conjectured and assessed that NLP techniques identifying equivalent requirements perform on a given dataset according to both ability and the odds of making correct identification. Also, they proposed a set of seven principles for evaluating the performance of NLP techniques in identifying equivalent requirements. They used IR methods such as Latent Semantic Analysis. Rago et al. [55] introduced a novel approach called ReqAligner that aids analysts to spot signs of duplication in use cases in an automated fashion. ReqAligner combines several text processing techniques, such as a use case classifier and a customized algorithm for sequence alignment.

Ambiguity is usually related to inconsistency. In the literature, many works have been proposed to tackle the problem of ambiguity in written requirements. Ferrari et al. [3] presented an NLP approach to identify ambiguous terms between different domains and rank them by ambiguity score. The approach is based on building domain-specific language models in each domain. They compared different word embeddings of one identical term from different domains to estimate its potential ambiguity across the domains of interest. There are some works using special terms and expressions with different POS or patterns [8]–[13]. Other works use heuristics to tackle coordination ambiguities (i.e., ambiguities brought by "and" or "or" conjunctions) [14] and anaphoric ones (i.e., ambiguities brought by pronouns) [15].

Our work complements to the existing researches in two aspects: 1) It is a method to resolving EC in RE. Detecting EC can improve the readability and understandability of requirements. 2) It is a deep learning approach, which is more powerful and generic.

B. Coreference Resolution

Our work is inspired by CDCR, so we review representative works on CR in recent years. For WDCR, Lee et al. [35] introduced the first end-to-end coreference resolution model without using a syntactic parser or handengineered mention detector. The key idea is to directly consider all spans in a document as potential mentions and learn distributions over possible antecedents for each. Joshi et al. [56] fine-tuned BERT to coreference resolution, achieving the state-of-the-art performance. However, they considered there is still room for improvement in modeling document-level context, conversations, and mention paraphrasing. As for CDCR, Lee et al. [57] introduced a novel coreference resolution system that models entities and events jointly by iteratively constructing clusters of entity and event mentions using linear regression to model cluster merge operations. The joint formulation allowed information from event coreference to help entity coreference, and vice versa. Inspired by [57], Barhom et al. [37] proposed a neural architecture for cross-document coreference resolution, which represents an event (entity) mention using its lexical span, surrounding context, and relation to entity (event) mentions via predicate-arguments structures.

CR presented in our work related to RE differs from NLP in two aspects: 1) Requirements in RE are domain-specific, so we cannot directly benefit from large general corpora or public knowledge bases. 2) In RE, the requirements and entities are related to a specific domain, where data annotation needs domain expertise and intensive manual effort. In our work, we use fine-tune technique and pre-training models to tackle these pivotal challenges.

IX. Conclusion and Future Work

This paper resolves entity coreference in requirement engineering. We propose a DEEP context-wise semantic method named DeepCoref for entity COREFerence detection. It consists of two parts: one is a fine-tuning BERT model for context representation, and the other is a Word2Vec-based network for entity representation. Then a multi-layer perceptron is followed to fuse and make a trade-off between two representations in order to obtain a better representation of the entity. We investigate the effectiveness of DeepCoref with 1853 samples on 21 projects from the industry community. The experimental results show that our approach significantly outperforms three baselines with average precision and recall of 96.10% and 96.06% respectively. In order to demonstrate the performance enhancement from different components, we compare DeepCoref with three variants as well.

In the future, we plan to add some event features into the entity representations based on what we have proposed in this work, because event information can help distinguish entities more precisely.

Acknowledgments

This work is supported by the National Key Research and Development Program of China under grant No.2018YFB140-3400, the National Science Foundation of China under grant No.61802374, No.61432001, No.61602450. This work is also supported by China Merchants Bank Intelligent Software Research and Development Effectiveness Research Project.

References

[1] M. E. C. Hull, K. Jackson, and J. Dick, *Requirements Engineering*. Springer London, 2002.

[2] X. Lian, M. Rahimi, J. Cleland-Huang, L. Zhang, R. Ferrai, and M. Smith, "Mining requirements knowledge from collections of domain documents," in *24th IEEE International Requirements Engineering Conference, RE 2016, Beijing, China, September 12-16, 2016*. IEEE Computer Society, 2016, pp. 156–165.

[3] A. Ferrari and A. Esuli, "An NLP approach for cross-domain ambiguity detection in requirements engineering," *Autom. Softw. Eng.*, vol. 26, no. 3, pp. 559–598, 2019.

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

[5] M. Mezghani, J. Kang, and F. Sèdes, "Industrial requirements classification for redundancy and inconsistency detection in SEMIOS," in *26th IEEE International Requirements Engineering Conference, RE 2018, Banff, AB, Canada, August 20-24, 2018*, G. Ruhe, W. Maalej, and D. Amyot, Eds. IEEE Computer Society, 2018, pp. 297–303.

[6] D. M. Fernández, S. Wagner, M. Kalinowski, M. Felderer, P. Mafra, A. Vetro, T. Conte, M. Christiansson, D. Greer, C. Lassenius, T. Männistö, M. Nayabi, M. Oivo, B. Penzenstadler, D. Pfahl, R. Prikladnicki, G. Ruhe, A. Schekelmann, S. Sen, R. O. Spínola, A. Tuzcu, J. L. de la Vara, and R. J. Wieringa, "Naming the pain in requirements engineering - contemporary problems, causes, and effects in practice," *Empirical Software Engineering*, vol. 22, no. 5, pp. 2298–2338, 2017.

[7] F. Fabbrini, M. Fusani, S. Gnesi, and G. Lami, "The linguistic approach to the natural language requirements quality: Benefits of the use of an automatic tool," in *Software Engineering Workshop, 2001. Proceedings. 26th Annual NASA Goddard*, 2001.

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

[9] S. F. Tjong and D. M. Berry, "The design of SREE - A prototype potential ambiguity finder for requirements specifications and lessons learned," in *Requirements Engineering: Foundation for Software Quality - 19th International Working Conference, REFSQ 2013, Essen, Germany, April 8-11, 2013. Proceedings*, ser. Lecture Notes in Computer Science, J. Dörr and A. L. Opdahl, Eds., vol. 7830. Springer, 2013, pp. 80–95.

[10] B. Gleich, O. Creighton, and L. Kof, "Ambiguity detection: Towards a tool explaining ambiguity sources," in *Requirements Engineering: Foundation for Software Quality, 16th International Working Conference, REFSQ 2010, Essen, Germany, June 30 - July 2, 2010. Proceedings*, ser. Lecture Notes in Computer Science, R. J. Wieringa and A. Persson, Eds., vol. 6182. Springer, 2010, pp. 218–232.

[11] B. Rosadini, A. Ferrari, G. Gori, A. Fantechi, S. Gnesi, I. Trotta, and S. Bacherini, "Using NLP to detect requirements defects: An industrial experience in the railway domain," in *Requirements Engineering: Foundation for Software Quality - 23rd International Working Conference, REFSQ 2017, Essen, Germany, February 27 - March 2, 2017, Proceedings*, ser. Lecture Notes in Computer Science, P. Grünbacher and A. Perini, Eds., vol. 10153. Springer, 2017, pp. 344–360.

[12] H. Femmer, J. Kucera, and A. Vetro, "On the impact of passive voice requirements on domain modelling," in *2014 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '14, Torino, Italy, September 18-19, 2014*, M. Morisio, T. Dybå, and M. Torchiano, Eds. ACM, 2014, pp. 21:1–21:4.

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

[14] F. Chantree, B. Nuseibeh, A. De Roeck, and A. Willis, "Identifying nocuous ambiguities in natural language requirements," in *14th IEEE International Requirements Engineering Conference (RE'06)*, 2006.

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

[16] P. Belsis, A. Koutoumanos, and C. Sgouropoulou, "PBURC: a patterns-based, unsupervised requirements clustering framework for distributed agile software development," *Requir. Eng.*, vol. 19, no. 2, pp. 213–225, 2014.

[17] D. Falessi, G. Cantone, and G. Canfora, "Empirical principles and an industrial case study in retrieving equivalent requirements via natural language processing techniques," *IEEE Trans. Software Eng.*, vol. 39, no. 1, pp. 18–44, 2013.

[18] W. W. Cohen, P. Ravikumar, and S. E. Fienberg, "A comparison of string distance metrics for name-matching tasks," in *Proceedings of IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03), August 9-10, 2003, Acapulco, Mexico*, S. Kambhampati and C. A. Knoblock, Eds., 2003, pp. 73–78.

[19] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*, 2010.

[20] C. Arora, M. Sabetzadeh, L. C. Briand, and F. Zimmer, "Automated extraction and clustering of requirements glossary terms," *IEEE Trans. Software Eng.*, vol. 43, no. 10, pp. 918–945, 2017.

[21] T. Gemkow, M. Conzelmann, K. Hartig, and A. Vogelsang, "Automatic glossary term extraction from large-scale requirements specifications," in *26th IEEE International Requirements Engineering Conference, RE 2018, Banff, AB, Canada, August 20-24, 2018*, 2018, pp. 412–417.

[22] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, 2013, pp. 3111–3119.

[23] J. P. Turian, L. Ratinov, and Y. Bengio, "Word representations: A simple and general method for semi-supervised learning," in *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden*, J. Hajic, S. Carberry, and S. Clark, Eds. The Association for Computer Linguistics, 2010, pp. 384–394.

[24] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2013.

[25] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, A. Moschitti, B. Pang, and W. Daelemans, Eds. ACL, 2014, pp. 1532–1543.

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

[27] X. Chen, Z. Liu, and M. Sun, "A unified model for word sense representation and disambiguation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, A. Moschitti, B. Pang, and W. Daelemans, Eds. ACL, 2014, pp. 1025–1035.

[28] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12, pp. 2493–2537, 2011.

[29] O. Melamud, D. McClosky, S. Patwardhan, and M. Bansal, "The role of context types and dimensionality in learning word embeddings," in *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, K. Knight, A. Nenkova, and O. Rambow, Eds. The Association for Computational Linguistics, 2016, pp. 1030–1040.

[30] O. Melamud, J. Goldberger, and I. Dagan, "context2vec: Learning generic context embedding with bidirectional LSTM," in *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, 2016, pp. 51–61.

[31] H. Palangi, L. Deng, Y. Shen, J. Gao, X. He, J. Chen, X. Song, and R. K. Ward, "Deep sentence embedding using the long short term memory network: Analysis and application to information retrieval," *CoRR*, vol. abs/1502.06922, 2015.

[32] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, ser. JMLR Workshop and Conference Proceedings, vol. 32. JMLR.org, 2014, pp. 1188–1196.

[33] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language*

Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), 2019, pp. 4171–4186.

[34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 5998–6008.

[35] K. Lee, L. He, M. Lewis, and L. Zettlemoyer, "End-to-end neural coreference resolution," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, M. Palmer, R. Hwa, and S. Riedel, Eds. Association for Computational Linguistics, 2017, pp. 188–197.

[36] G. R. Doddington, A. Mitchell, M. A. Przybocki, L. A. Ramshaw, S. M. Strassel, and R. M. Weischedel, "The automatic content extraction (ACE) program - tasks, data, and evaluation," in *Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC 2004, May 26-28, 2004, Lisbon, Portugal*. European Language Resources Association, 2004.

[37] S. Barhom, V. Shwartz, A. Eirew, M. Bugert, N. Reimers, and I. Dagan, "Revisiting joint modeling of cross-document entity and event coreference resolution," in *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, A. Korhonen, D. R. Traum, and L. Màrquez, Eds. Association for Computational Linguistics, 2019, pp. 4179–4189.

[38] S. Beheshti, B. Benatallah, S. Venugopal, S. H. Ryu, H. R. Motahari-Nezhad, and W. Wang, "A systematic review and comparative analysis of cross-document coreference resolution methods and tools," *Computing*, vol. 99, no. 4, pp. 313–349, 2017.

[39] J. Lu and V. Ng, "Joint learning for event coreference resolution," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, R. Barzilay and M. Kan, Eds. Association for Computational Linguistics, 2017, pp. 90–101.

[40] T. Johann, C. Stanik, A. M. A. B., and W. Maalej, "SAFE: A simple approach for feature extraction from app descriptions and app reviews," in *25th IEEE International Requirements Engineering Conference, RE 2017, Lisbon, Portugal, September 4-8, 2017*, 2017, pp. 21–30.

[41] E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng, "Improving word representations via global context and multiple word prototypes," in *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 1: Long Papers*, 2012, pp. 873–882.

[42] L. Logeswaran, M. Chang, K. Lee, K. Toutanova, J. Devlin, and H. Lee, "Zero-shot entity linking by reading entity descriptions," in *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, A. Korhonen, D. R. Traum, and L. Màrquez, Eds. Association for Computational Linguistics, 2019, pp. 3449–3460.

[43] S. Li, Z. Zhao, R. Hu, W. Li, T. Liu, and X. Du, "Analogical reasoning on chinese morphological and semantic relations," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics,*

[47] A. Marcus and J. I. Maletic, "Identification of high-level concept clones in source code," in *16th IEEE International Conference on Automated Software Engineering (ASE 2001), 26-29 November 2001, Coronado Island, San Diego, CA, USA*, 2001, pp. 107–114.

ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers, 2018, pp. 138–143.

[44] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[45] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[46] J. I. Maletic and A. Marcus, "Using latent semantic analysis to identify similarities in source code to support program understanding," in *12th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2000), 13-15 November 2000, Vancouver, BC, Canada*, 2000, pp. 46–53.

[48] ——, "Recovering documentation-to-source-code traceability links using latent semantic indexing," in *Proceedings of the 25th International Conference on Software Engineering, May 3-10, 2003, Portland, Oregon, USA*, 2003, pp. 125–137.

[49] A. Mahmoud and N. Niu, "On the role of semantics in automated requirements tracing," *Requir. Eng.*, vol. 20, no. 3, pp. 281–300, 2015.

[50] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, "Indexing by latent semantic analysis," *JASIS*, vol. 41, no. 6, pp. 391–407, 1990.

[51] W. Wang, N. Niu, H. Liu, and Z. Niu, "Enhancing automated requirements traceability by resolving polysemy," in *26th IEEE International Requirements Engineering Conference, RE 2018, Banff, AB, Canada, August 20-24, 2018*, 2018, pp. 40–51.

[52] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes*, 1995, pp. 1137–1145.

[53] W. Gomaa and A. A. Fahmy, "A survey of text similarity approaches," *International Journal of Computer Applications*, vol. 68, no. 13, 2013.

[54] E. Jürgens, F. Deissenboeck, M. Feilkas, B. Hummel, B. Schätz, S. Wagner, C. Domann, and J. Streit, "Can clone detection support quality assessments of requirements specifications?" in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2, ICSE 2010, Cape Town, South Africa, 1-8 May 2010*, J. Kramer, J. Bishop, P. T. Devanbu, and S. Uchitel, Eds. ACM, 2010, pp. 79–88.

[55] A. Rago, C. A. Marcos, and J. A. Diaz-Pace, "Identifying duplicate functionality in textual use cases by aligning semantic actions," *Software and Systems Modeling*, vol. 15, no. 2, pp. 579–603, 2016.

[56] M. Joshi, O. Levy, L. Zettlemoyer, and D. S. Weld, "BERT for coreference resolution: Baselines and analysis," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds. Association for Computational Linguistics, 2019, pp. 5802–5807.

[57] H. Lee, M. Recasens, A. X. Chang, M. Surdeanu, and D. Jurafsky, "Joint entity and event coreference resolution across documents," in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, J. Tsujii, J. Henderson, and M. Pasca, Eds. ACL, 2012, pp. 489–500.