Requirements Engineering (2022) 27:351–373
https://doi.org/10.1007/s00766-022-00374-8

ORIGINAL ARTICLE

Detecting coreferent entities in natural language requirements

Yawen Wang[1,2] · Lin Shi[1,2] · Mingyang Li[1,2] · Qing Wang[1,2,3,4] · Yun Yang[5]

Received: 18 December 2020 / Accepted: 21 January 2022 / Published online: 1 March 2022

Abstract
Requirements are usually written in natural language and evolve continuously during the process of software development, which involves a large number of stakeholders. Stakeholders with diverse backgrounds and skills might refer to the same real-world entity with different linguistic expressions in the natural-language requirements, resulting in requirement inconsistency. We define this phenomenon as Entity Coreference (EC) in the Requirement Engineering (RE) area. It can lead to misconception about technical terminologies, and harm the readability and long-term maintainability of the requirements. In this paper, we propose a DEEP context-wise method for entity COREFerence detection, named DeepCoref. First, we truncate corresponding contexts surrounding entities. Then, we construct a deep context-wise neural network for coreference classification. The network consists of one fine-tuning BERT model for context representation, a Word2Vec-based network for entity representation, and a multi-layer perceptron in the end to fuse and make a trade-off between two representations. Finally, we cluster and normalize coreferent entities. We evaluate our method, respectively, on coreference classification and clustering with 1853 industry data on 21 projects. The former evaluation shows that DeepCoref outperforms three baselines with average precision and recall of 96.10% and 96.06%, respectively. The latter evaluation on six metrics shows that DeepCoref can cluster coreferent entities more accurately. We also conduct ablation experiments with three variants to demonstrate the performance enhancement brought by different components of neural network designed for coreference classification.

Keywords Entity coreference · Requirements inconsistency · Deep learning · Requirement engineering

1 Introduction

In the stage of conception, requirement specifications are specified in natural language with the flexibility to accommodate the arbitrary abstraction [30]. Most requirements are written by different stakeholders with diverse backgrounds and skills [9, 21, 45]. Writing requirements clearly without inconsistency and ambiguity before passing to the subsequent stages of the development is a challenging but essential task [20, 56]. The inconsistency violates one of the quality principles related to linguistic aspects of natural-language requirements [16]. It might occur among requirement analysts and domain experts because of their specialized jargons, or stakeholders from different domains [21].

In practice, different linguistic expressions could be used by different stakeholders to refer to the same real-world entity in natural-language requirements, and we define such phenomena as Entity Coreference (EC). More specifically, we present an example of EC in Fig. 1 for illustration. Suppose we have three pieces of natural-language requirement texts (R1, R2, and R3) and their related entities:

✉ Lin Shi
shilin@iscas.ac.cn

Yawen Wang
yawen2018@iscas.ac.cn

Mingyang Li
mingyang@itechs.iscas.ac.cn

Qing Wang
wq@iscas.ac.cn

Yun Yang
yyang@swin.edu.au

1 Laboratory for Internet Software Technologies, Institute of Software, Chinese Academy of Sciences, Beijing, China

2 University of Chinese Academy of Sciences, Beijing, China

3 State Key Laboratory of Computer Sciences, Institute of Software, Chinese Academy of Sciences, Beijing, China

4 Science & Technology on Integrated Infomation System Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing, China

5 School of Software and Electrical Engineering, Swinburne University of Technology, Melbourne, Australia

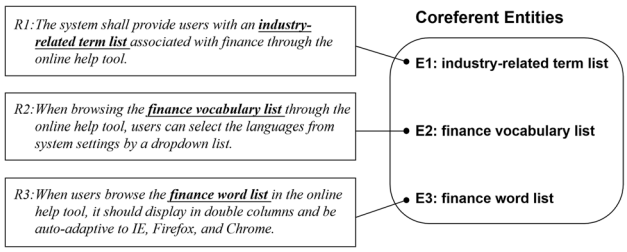


Fig. 1 Examples of coreferent entities in natural-language requirements, which make the requirements difficult to understand

"industry-related term list" in R1, "finance vocabulary list" in R2 and "finance word list" in R3. However, we can conclude the three entities refer to the same thing according to their contexts. EC might lead to misconception on entities, thus impairing the readability and understandability of requirements [60]. This work focuses on resolving EC in Requirement Engineering (RE)[1].

To tackle the problem of inconsistency or ambiguity in natural-language requirements, researchers have proposed many works literally. We classify these works into three categories. Pattern-based methods use some special terms and expressions of different Part-of-Speech (PoS) and other patterns [6, 18, 19, 23, 67, 72] for inconsistency detection, or heuristics to tackle coordination or anaphoric ambiguities [7, 78]. Learning-based methods [5, 17, 56] use information retrieval (IR) techniques such as Latent Semantic Indexing (LSI) or unsupervised clustering algorithms such as K-Means. Similarity-based methods include word embeddings [21] and syntactic methods (e.g., Jaccard [10] and Levenstein [51]) by computing a similarity score between entities. However, these methods cannot be directly utilized in EC due to the following challenges:

- *Multi-word entity* In natural-language requirements, most entities are noun phrases [1, 22] rather than a single word. For example, all entities shown in Fig. 1 consist of multiple words. On average, each entity contains 3.52 words based on observations on our industry data. However, it is challenging to represent multi-word entities with word-level representation techniques. Take the entities in Fig. 1 as an example, the expression of entity E1 is quite different from the expressions of the other two entities E2 and E3, that they only share one identical word "list". However, E1 refers to the same entity as E2 and E3. If we simply use the word-level similarity methods such as word embedding, incorrect EC will be given that E2 and E3 are coreferent entities while E1 is a different one.
- *Missing contextual semantics* Sentence-level contextual semantic information can provide extra information for resolving EC, which is ignored by existing works. In most cases, we infer whether two entities are coreferent based on their contexts, because coreferent entities usually have similar contexts. For example in Fig. 1, similar contextual words such as "user" and "online help tool" appear in all the three requirements, which indicates three entities are coreferent. Therefore, add and how to add contextual semantics into entity representations is important as well.
- *Insufficient annotated resources* Domain expertise and intensive manual effort are required when annotating coreferent entities in requirements, resulting in insufficient annotated data for effective learning. In addition, EC detection in RE is a domain-specific task. It cannot directly benefit from large general corpora or public knowledge bases like general coreference detection tasks. How to use limited annotation data and benefit from pre-trained models trained on large general corpora is another challenge.

Based on the challenges addressed above, our previous work [76] proposed a **DEEP** context-wise semantic method to resolving entity **COREF**erence in natural-language requirements, which is named DeepCoref. It first performs **Context Truncation** to truncate context for each entity and then convert $\langle context, entity \rangle$ pairs to model input format. Then, it performs **Coreference Classification** to infer whether two entities are coreferent by constructing a context-wise coreference network. 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. Subsequently, 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 a predicted label to infer whether two entities are coreferent.

However, DeepCoref has one major limitation that it can only resolve coreference between entity pairs, which seriously affected its practicality in requirement analysis activities, such as establishing a non-coreferent entity dictionary. Establishing and maintaining such a dictionary, which characterizes the key functional objects of the software system, is an effective mechanism to tackle issues triggered by requirement evolution. These issues include ambiguous or duplicate features, lacking of visibility in requirement dependency, poor scope and cost estimation of software projects [42]. The non-coreferent entity dictionary can also support new feature identification [31], requirement change analysis [34], and effort estimation based on requirement changes [35], etc.

[1] Note that entities in this research are ready-made, and entity extraction is out of the scope of this research.

To support requirement analysts addressing these issues, we enhance DeepCoref by integrating two new components:

- *Clustering* is to cluster all coreferent entities based on coreference relations of entity pairs, so that coreferent entities are assigned into the same clusters and non-coreferent ones into different clusters.
- *Normalization* is to select one from all coreferent entities, so that each cluster of coreferent entities can have one normalized name.

Clustering can link all coreferent entity pairs together, which help requirement analysts understand which entities are coreferent to the same conception. Normalization can assign a normalized name for coreferent entities, which help requirement analysts reduce misconception caused by different expressions.

We investigate the effectiveness of DeepCoref with data from our industry partner. The experimental results of coreference classification show that our method outperforms three baselines, with average precision and recall of 96.10% and 96.06%, respectively. The clustering performance of DeepCoref are higher as well than two baselines on all the six clustering evaluation metrics. We also conduct ablation experiments for network design of DeepCoref with three variants to demonstrate the performance enhancement brought by different components in coreference classification.

The main contributions of this paper are as follows:

- We highlight the importance of detecting entity coreference in RE.
- We propose a deep context-wise coreference network which combines contextual semantics for automatic coreference classification, a method of clustering coreferent entities, and a method of normalizing the expressions of coreferent entities.
- From the perspective of coreference classification and clustering, we conduct experimental evaluation on 1853 samples of 21 projects from the industry community with promising results.
- Public-access of source code[2] to facilitate the replication of our study and its application in other contexts.

The rest of the paper is organized as follows. Section 2 describes the background. Section 3 presents the design of our proposed method. Sections 4 and 5 show the experimental setup and evaluation results, respectively. Section 6 provides a detailed discussion. Section 7 describes threats to validity. Section 8 surveys related work. Finally, we summarize the paper in Sect. 9.

2 Background

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

2.1 Fine-tuning BERT

BERT (Bidirectional Encoder Representations from Transformers) [13] is a deep bidirectional *Transformer* encoder [74] trained with the objective of masked language modeling and the next-sentence prediction task, which proves effective in various NLP tasks. It is constructed based on *Transformer* architecture [74], which is proposed to use a *stacked self-attention* encoder-decoder structure to replace conventional LSTM [27] architecture. It also introduces *multi-headed attention* to improve previous attention mechanisms, which helps the overall model to focus on different positions and solves the problem that the current word itself can dominate other words [26, 74]. Radford et al. [64] introduce the concepts of the *Transformer* architecture that can be fine-tuned. They verify that large performance enhancement can be realized by generative pre-training of a language model on a diverse corpus of unlabeled text, followed by discriminative fine-tuning on each specific task.

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=110 M) and $BERT_{LARGE}$ (L=24, H=1024, A=16, Total Parameters=340 M), 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.

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

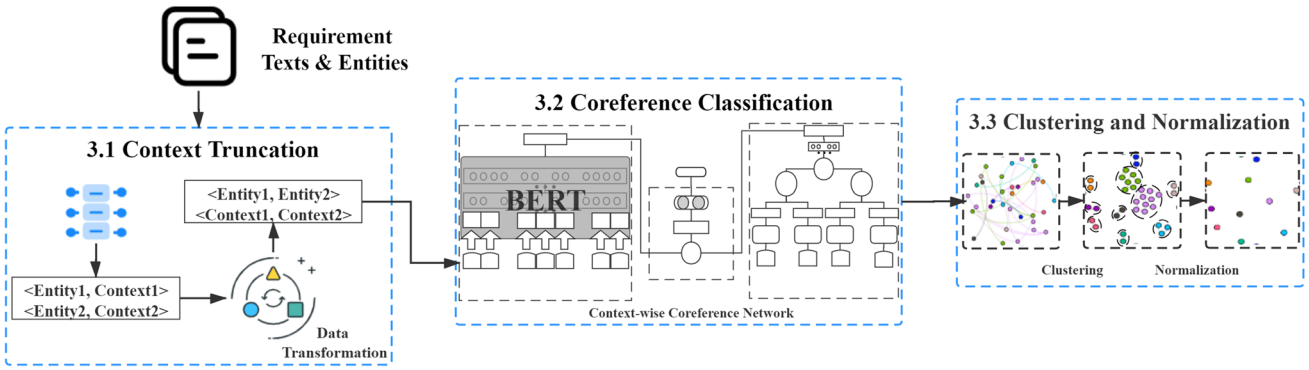


Fig. 2 The overview of DeepCoref

2.2 Word embeddings

Embedding (also known as distributed representation [59, 73]) 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 [58, 59]. 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 [58, 59, 63]. Comparing with Levenstein [51], here "similar" means semantic similarity instead of string similarity. Word embeddings are based on the distributional hypothesis of Harris [25]. 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 [58]. 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 [59]. *word2vec* [58] introduced by Mikolov et al. is the most typical method. Another word embedding model is GloVe [63], 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 [8], coreference resolution and named entity recognition (NER) [11, 55, 73]. 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 [54]. Likewise, a sentence (i.e., a sequence of words) can also be embedded as a vector [62]. A simple way of sentence embedding is, for example, to consider it as a bag of words and add up all its word vectors [39].

2.3 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 [2, 4, 14, 41]. 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 [14].

There are two types of tasks in CR [2]: (1) resolving coreference of entities or 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 [2, 47]. 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 [2]. WDCR methods provide techniques for the identification of mentions in one document that refer to the same underlying entity/event, while CDCR methods provide techniques for the identification of mentions in different documents [4]. This work is most inspired by the CDCR entity methods, but at the same time, it is revised for tackling the particularity of EC in the context of RE. We also list some differences between EC in RE and EC in general NLP contexts in Sect. 8.2.

3 Approach

To address the challenges mentioned in Sect. 1, we propose a method named DeepCoref for resolving EC detection. Figure 2 presents the overview of DeepCoref. Given a set of requirement texts written in natural language and its related entities, we firstly truncate their corresponding contexts (see Sect. 3.1). Then, we build a context-wise coreference network (see Sect. 3.2) for coreference classification. The network can predict whether a pair of entities are semantically equivalent, and the output is the predicted label (1 for coreference and 0 for non-coreference). Finally, we cluster and normalize all coreferent entities (see Sect. 3.3) according to the predicted results among entity pairs.

3.1 Context truncation

Since entity extraction has been widely developed by many NLP researches [1, 22, 31, 42, 70], 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 provided from our industry partner.

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. The fixed window size can also avoid too long texts and align text sequences of different length. 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 (i.e., 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 $l_{pre} < \lceil \frac{M-N}{2} \rceil$, all previous words are reserved, and the subsequent text sequence is truncated by length $min(l_{sub}, M - N - l_{pre})$, where $min(\cdot)$ is to take the minimum.

The final extracted context is a concatenation of truncated previous sequence (denoted as *pre*), the entity itself (denoted as *entity*) 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. In this work, we set window size $M = 128$.

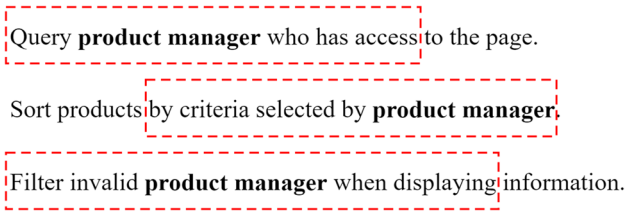


Fig. 3 An example of context truncation. The bold words (e.g., product manager) are entity words. The red dotted rectangle represents the window. Here window size $M = 6$, and the length of entity $N = 2$

Figure 3 demonstrates an example of context extraction for each case. By context truncation, we obtain the entity and its related context (i.e., $\langle context, entity \rangle$). Finally, we perform data transformation to format two $\langle context, entity \rangle$ pairs into a context pair (i.e., $\langle context_1, context_2 \rangle$, and a entity pair (i.e., $\langle entity_1, entity_2 \rangle$).

3.2 Coreference classification

We build a context-wise coreference network for coreference classification between two entities. The architecture of the network are shown in Fig. 4. The context-wise coreference network takes a pair of entities and their related contexts as input and predicts whether two entities are coreferent. The network consists of two parts. One is a fine-tuning BERT model for learning context representations, and the other 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 to fuse two representations, and a softmax layer to infer the predicted labels.

3.2.1 Fine-tuning BERT context model

A powerful context representation is helpful for measuring context-wise similarity [28]. In many NLP tasks (e.g., entity disambiguation and entity/event 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 methods do not include any mechanism for optimizing the representation of the entire contextual sentences [54].

To obtain a good context representation, we use BERT which is a fine-tuning based and bidirectional pre-training representation model [13]. It takes a context pair (i.e., $\langle context_1, context_2 \rangle$) as input, and produces a context vector representation. Due to the contexts are usually short text, we use the model $BERT_{BASE}$ with a relatively small model size,

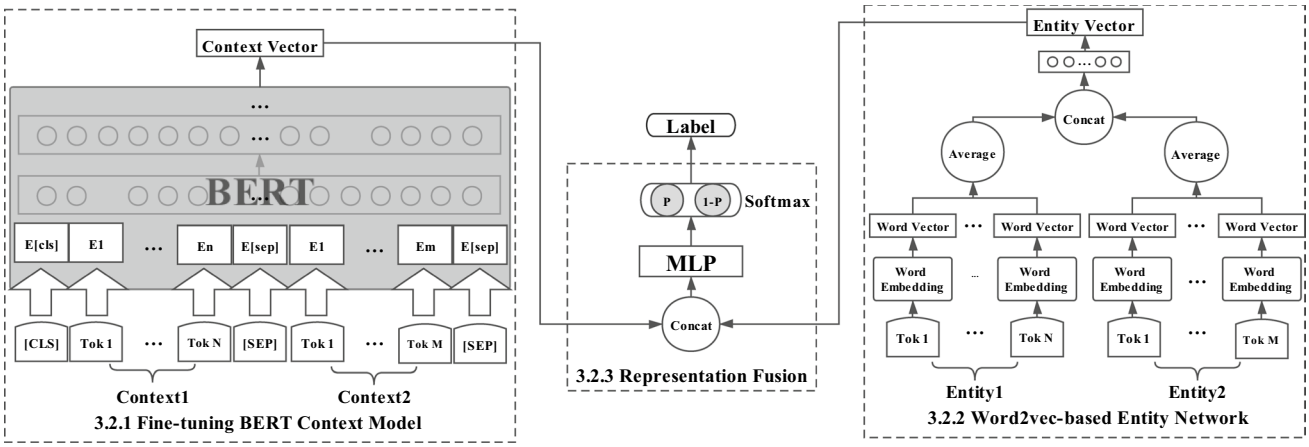


Fig. 4 The architecture of context-wise coreference network

which has 12 layers, 768 hidden dimensions and 12 attention heads. In BERT, the input can be a pair of sentences. 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*] [13, 46].

3.2.2 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 [59]. It takes an entity pair (i.e., $\langle entity_1, entity_2 \rangle$) as input, and produces an entity vector representation. We utilize the 300 dimensional word embeddings which are pre-trained on a 1.3G Wikipedia corpus[3] with 223M tokens and 2129K vocabularies. It is trained with three types of features (word features, n-gram features and character features) using the skip-gram model with negative sampling [44].

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 e). So the entity pair can be represented as a vector (denoted as p) which is concatenated by the embeddings of two entities ($p = [e_1 \oplus e_2]$). Since the dimension of word embeddings is 300, the dimension of e is 300 and the dimension of p is 600. After that, p is fed into a fully connected layer to produce an entity vector representation (denoted as v_t).

3.2.3 Representation fusion

The output of two parts of context-wise coreference 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.

Following previous practice of representation fusion [2], 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 [71] 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 $[s_1, s_2]$ 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:

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

Then $[s_1, s_2]$ can be normalized to probabilities $[p, 1-p]$, where $p \in [0, 1]$. The network can infer the predicted label based on these probabilities.

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

3.2.4 Training details and implementation

Training details: Since the task is a classification problem, we use cross-entropy as the loss function, which is specified as:

$$Loss = \sum_{x} p(x) \cdot log(\frac{1}{q(x)}) \tag{2}$$

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

The design of the context-wise coreference network makes all parameters jointly fine-tuned on a specific task (i.e., coreference classification), which can benefit from large corpora pre-training in a relatively inexpensive way. It also alleviates insufficient annotated resource problem 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.

Implementation: We implement context-wise coreference network using Transformers[4] [77] which is an open-source library for natural language understanding and natural language generation with over 32+ pre-trained models built on Pytorch[5].

3.3 Clustering and normalization

Given a pair of entities and their corresponding contexts, we can use our trained context-wise coreference network to predict whether two entities are coreferent. Based on the coreference relations of all entity pairs, we can establish a non-coreferent entity dictionary by two steps: clustering and normalization.

3.3.1 Clustering

Algorithm 1 Clustering

Input: $d = List\ ((e_i, e_j, p))$, $e = List\ (e_i)$, r.
$e_i, e_j \in e$: node (entity), p: weight (probability),
d: edges, e: node (entity) list, r:ratio.
Output: $c = Map\ \{e_i : c_i\}$.
$e_i \in e$: node (entity), c_i: cluster id, c: e_i-c_i map.

Initialize: $cluster = 0$, $c = Map\{e_i : null\}$
Update d: Remove (e_i, e_j, p) **if** $p < r$
for each Tuple (e_i, e_j, p) **in** d **do**
  if $c[e_i] == null$ and $c[e_j] == null$ **then**
    $c[e_i] = cluster$, $c[e_j] = cluster$
    $cluster+ = 1$
  end if
  if $c[e_i] != null$ and $c[e_j] == null$ **then**
    $c[e_j] = c[e_i]$
  end if
  if $c[e_i] == null$ and $c[e_j] != null$ **then**
    $c[e_i] = c[e_j]$
  end if
  if $c[e_i] != null$ and $c[e_j] != null$ **then**
    if $c[e_i] < c[e_j]$ **then**
      for each e_n **in** c **do**
        if $c[e_n] == c[e_j]$ **do** $c[e_n] = c[e_i]$
      end for
    end if
    if $c[e_i] > c[e_j]$ **then**
      for each e_n **in** c **do**
        if $c[e_n] == c[e_i]$ **do** $c[e_n] = c[e_j]$
      end for
    end if
  end if
end for
for each e_n **in** c **do**
  if $c[e_n] == null$ **then**
    $c[e_n] = cluster$
    $cluster+ = 1$
  end if
end for
return c

In this step, we cluster all the entities that are coreferent to the same concept into one cluster. Thus, different clusters will indicate different concepts with all their coreferent entities. We define EC in RE has two properties: *Symmetry* and *Transitivity*. Let E be the entity set, then

- *Symmetry*: For $e_i, e_j \in E$, if e_i is coreferent to e_j, then e_j is coreferent to e_i.
- *Transitivity*: For $e_i, e_j, e_k \in E$, if e_i is coreferent to e_j, and e_j is coreferent to e_k, then e_i is coreferent to e_k.

Ideally, each entity can be assigned into some cluster based on predicted results of entity pairs following these two properties. However, one challenge is that there might be conflicts among predicted results of entity pairs. For example,

[4] https://github.com/huggingface/transformers.
[5] https://pytorch.org.

if entity A is predicted coreferent to entity B, and entity B is predicted coreferent to entity C, then entity A should be predicted coreferent to entity C, and three entities should be assigned into one cluster. However, there exists one situation that DeepCoref wrongly predicts entity A and entity C as non-coreferent. Such conflicts are caused by misclassification in predicted results. Therefore, we construct a clustering algorithm to cluster all entities based on the predicted results of entity pairs and resolve conflicts simultaneously.

Concretely, we build a graph, where we treat all entities as nodes, and coreference relations between entities as edges. We take the probability of two entities being predicted coreferent by coreference network, as the weights of edges. Each edge can be denoted as $\langle e_i, e_j, p\rangle$, where e_i and e_j are entities (nodes), and p ($p \in [0, 1]$) indicates the probability of entity e_i and entity e_j being coreferent. To avoid the conflicts, we first remove edges whose weights p are less than a ratio r (line 2). The ratio r is an empirical value, which is set as 0.7 in this work. It indicates that only two entities whose probability of being coreferent is over 0.7 would be assigned into one cluster, because the lower predicted confidence might lead to conflicts. Then, we traverse all edges (line 3-26). If neither e_i and e_j have been assigned a cluster id, we assign a new one to them (line 4-7). If one of e_i and e_j has been assigned a cluster id, we assign this id to the one without an id (line 8-13). When both e_i and e_j have been assigned a cluster id, if they belong to different clusters, we need to merge the cluster with the higher id into the cluster with the lower id (line 14-25). If they belong to the same cluster, we skip it. Finally, we assign each entity, which is not assigned a cluster id after traversal, a new cluster id (line 27-32). More details can be seen in Algorithm 1.

3.3.2 Normalization

After clustering, we normalize the entities by selecting one of coreferent entities belonging to the same cluster. Thus, all coreferent entities in the same cluster are normalized into one name. More specifically, for each cluster, we apply TextRank algorithm [57] to construct a weighted word graph (denoted as $G = (V, E)$) based on the word co-occurrence relations in the contextual texts of each cluster. The nodes V in the graph represents the words in the context, and the edges E in the graph represents the co-occurrence relations between word V_i and its neighboring word V_j. The TextRank score of node V_i is specified as:

$$S(V_i) = (1 - d) + d * \sum_{V_j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j) \tag{3}$$

where $In(V_i)$ is the node set that points to the node V_i, and $Out(V_j)$ is the set of nodes pointed to by the node V_j. $|\cdot|$ is the number of the nodes. d is the damping coefficient, and the value 0.85 is taken generally. The bigger score represents the more "importance" of the vertex within the graph. Then for each coreferent entity in the same cluster, we perform word segmentation, and calculate the TextRank score of each word. We treat the average score as the score of the whole entity, and select the entity with the highest score as the normalized entity.

4 Experiment design

4.1 Research questions

Our evaluation addresses the following four research questions.

RQ1 (Coreference classification) How effective is the coreference classification of DeepCoref *compared with existing techniques?* To investigate the effectiveness of coreference classification of our method, we conduct 10-fold cross-validation using data from our industry partner. We compare the performances of three baselines (see Sect. 4.3.1). These methods include syntactic or semantic similarity measures to detect coreference on word level or sentence level. We compare these methods to demonstrate the advantage of combining entity and context representations. Besides, we also present statistical results by the project to examine the stability and generalizability across different projects.

RQ2 (Clustering and normalization) How effective is the clustering based on the predicted results of entity pairs derived from coreference classification? In RQ2, we investigate the performance of clustering from three perspectives.

- *Clustering evaluation:* To verify the impact of different coreference techniques (i.e., Word2Vec, LSI and Levenstein) on the clustering performance, we perform the algorithm introduced in Sect. 3.3.1 on coreferent entity pairs obtained by DeepCoref and three baseline coreference techniques, and compare their clustering performance. To demonstrate the effectiveness of our proposed clustering method, we compare the performance of DeepCoref with two commonly used clustering algorithms (see Sect. 4.3.2). We use six metrics to evaluate the clustering performance by comparing the ground-truth and predicted cluster labels.
- *Example analysis* To present the clustering results more intuitively, we give network graphs to illustrate the differences of clusters among DeepCoref and two clustering baselines. In addition, we present the examples of coreferent/non-coreferent entities of clusters in network graphs.

- *Classification improvement* To verify the effectiveness of our proposed clustering method on resolving conflicts caused by misclassification, we also present the performance improvement on coreference classification brought by clustering.

RQ3 (Ablation experiment) How effectively does each component of context-wise coreference network facilitate coreference classification? To examine the performance enhancement introduced by context representations and entity representations, respectively, we construct three variants (see Sect. 4.4): DeepCoref-ctx, DeepCoref-entity and DeepCoref-LSI. We conduct 10-fold cross-validation on DeepCoref and three variants, respectively, to demonstrate the effectiveness for combining two representations.

RQ4 (Data sensitivity) To what degree does data size influence coreference classification results? In RQ4, we conduct an experiment by increasingly enlarging the size of training data to examine the sensitivity between performance enhancement and data augmentation. The amount of data verifies whether fine-tuning methods can alleviate low-resource problem. We also give the time consumption with the increase of data.

4.2 Data preparation

Our experimental data are collected from the repositories of China Merchants Bank (CMB)[6]. We collaborate with the project management department of CMB and 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 with the support of automated tools, audited by project management department and well-maintained through the requirement evolution. We prepare data in the following steps: *Pre-processing*, *Sampling* and *Ground-truth Labeling*.

4.2.1 Pre-processing

For each entity and its related requirement text, we filter noisy tokens such as URL, HTML tags and SQL statements with Ekphrasis[7] [3] which is a collection of light-weight text processing tools. These tokens are produced by their management system but not removed when dumped from the system. Then, we filter some template words (e.g., "I would like to") which cannot contribute to the result but introduce noise, especially for contextual semantic similarity. As the BERT language model has its own vocabulary and pre-processing steps, we do not perform other pre-processing to deal with punctuations and specific tokens.

4.2.2 Sampling

After pre-processing, we truncate the context for each entity, which is demonstrated in Sect. 3.1. 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* (i.e., $\langle ctx\text{-}entity_1, ctx\text{-}entity_2 \rangle$) from the same project in a combination way. The combination step is to build relations among $\langle context, entity \rangle$ pairs for ground-truth labeling.

4.2.3 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 three steps:

- The project management department of CMB assigns samples (i.e., $\langle ctx\text{-}entity_1, ctx\text{-}entity_2 \rangle$ 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.
- The labeling results withdrawn from each project team are reviewed by the project management department.
- As for samples which are annotated to different labels, two teams would discuss and decide through voting. Only those samples where both teams make a full agreement can be included in our dataset.

In addition, the labeling process strictly follows the two properties (symmetry and transitivity) of coreference relation to avoid inconsistency.

Finally, we totally collect 5,736 labeled data, and the agreement between the labeling results from project team and the results reviewed by project management team takes up 5,083 (88.62%). The Cohen's Kappa reaches 0.77. After discussion and voting, common consensus is reached for every entity pair. The two classes are extremely imbalanced (i.e., 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 ($\langle ctx\text{-}entity_1, ctx\text{-}entity_2, label \rangle$). The positive labels (897, 48.41%) mean $ctx\text{-}entity_1$ and $ctx\text{-}entity_2$ are coreferent, negative labels (956, 51.59%) for non-coreference. The total number of entities in our dataset is 639. These entities form 355 clusters, and each cluster contains entities which are coreferent to the same concept. This indicates that there exists many small clusters,

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

[7] https://github.com/cbaziotis/ekphrasis.

even some clusters only contain one single entity (i.e., no other coreferent entities).

4.3 Baselines

To further demonstrate the advantages of DeepCoref, we compare it with three commonly used techniques for coreference classification, and two commonly used methods for clustering.

4.3.1 Baselines for coreference classification

Word2Vec: Ferrari et al. [21] present a method 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 multiple words. We use an average of word embeddings to represent an entity, and then compute a cosine 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 method 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 [49], detecting high-level conceptual code clones [52], and recovering traceability links between documentation and source code [53], which is considered to be able to resolve the polysemy problem as well [12, 48, 75]. 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 (i.e., deletion, insertion and substitution) to convert one entity into the other [1, 51]. We use the implementation in library Distance[8].

4.3.2 Baselines for clustering

K-Means: It is a commonly used clustering algorithm. To compare with our proposed clustering algorithm, we vectorize entities following the idea of DeepCoref. Concretely, we first encode each entity with word embeddings, and encode corresponding contexts with BERT. Then, we concatenate the two vectors to represent the entities. Finally, we run K-Means on these vectors to assign all entities into clusters.

DBSCAN [15]: It is a density-based algorithm for discovering clusters of arbitrary shape. Similarly, we encode entities with word embeddings and contexts with BERT, and then concatenate the two vectors to represent entities. Finally, we run DBSCAN to cluster all entities.

We implement the two clustering algorithm with the library scikit-learn[9], and apply the optimal hyper-parameters after tuning for the best performance.

4.4 Experimental setup

For hyper-parameter settings, the learning rate is set 10^{-5}. The optimizer is Adam [36] algorithm. We use the mini-batch technique [43] 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. These hyper-parameter are tuned carefully for best performance, and keep unchanged across all experiments.

We conduct 10-fold cross-validation [37] on the dataset collected from CMB in RQ1 and RQ3. 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 RQ2, for each coreference technique, we keep the predicted results of each fold, then we can obtain the predicted results of the whole dataset by merging 10-fold results. After that, we construct predicted clusters of each coreference technique by performing Algorithm 1. For clustering baselines, We apply K-Means and DBSCAN to the combined vector representations of entities and contexts to obtain predicted clusters. In RQ4, we randomly split data into the training set (90%) and testing set (10%), and keep the testing set unchanged. Then we enlarge the size of training set, and evaluate on the fixed testing set.

RQ1 (Coreference classification): RQ1 is to question the effectiveness of coreference classification of DeepCoref comparing with existing techniques. In this experiment, we compare the performance of DeepCoref with three methods (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 methods. The output of Levenstein is a similarity score. The outputs of Word2Vec and LSI are vector representations, and

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

[9] https://scikit-learn.org.

subsequently, we infer the predicted label by computing similarity score via cosine similarity [24]. 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 (see details for ratio selection in Sect. 6.1). In addition, we give statistical results by the project to examine the stability and generalizability across different projects.

RQ2 (Clustering and normalization): RQ2 is to question the effectiveness of clustering of DeepCoref.

- *Clustering evaluation:* We perform Algorithm 1 on each coreference technique (i.e., Word2Vec, LSI and Levenstein), and use six metrics (see Sect. 4.5) to evaluate the performance of clustering. Note that in order to perform Algorithm 1 on Word2Vec, LSI and Levenstein, we simply treat the predicted labels as the probability of two entities being predicted coreferent (i.e., label 1 is 100%, and label 0 is 0%). We also compare the performance of DeepCoref and two clustering baselines (i.e., K-Means and DBSCAN) on six metrics. More specifically, we encode entities with word embeddings and corresponding contexts with BERT, and concatenate them as vector representations of entities. Then we perform K-Means and DBSCAN on these vectors to cluster entities.
- *Example analysis:* For a more intuitive illustration, we also present network graphs for clusters of the ground-truth, DeepCoref and two clustering baselines, respectively. For clustering results obtained by each method, we plot the clusters with network graphs, where nodes denote entities. The entities (nodes) belonging to the same cluster are gathered together. We use different color to distinguish different clusters according to the cluster labels in the ground-truth. The entities with the same color belongs to the same cluster in the ground-truth. We only color top eight clusters for the ground-truth and each method, and other relatively small clusters are all painted grey, because we totally have 355 clusters, and adding more clusters with few entities in the graph would reduce clarity. In addition, we present the examples of coreferent/non-coreferent entities according to clusters in the network graph.
- *Classification improvement:* To investigate the effectiveness of our proposed clustering method on resolving conflicts caused by misclassification, we retrieve the coreference classification results of entity pairs based on the clustering results, and compare with the ground-truth labels.

RQ3 (Ablation experiment): RQ3 is to question the contribution of each component of DeepCoref to the performance enhancement of coreference classification. We demonstrate the performance enhancement introduced by each component of context-wise coreference network by constructing DeepCoref *-ctx* and DeepCoref *-entity*. DeepCoref-ctx only contains the fine-tuning BERT model for context representations without the Word2Vec-based network for entity representations, and DeepCoref-entity is totally opposite to DeepCoref-ctx only with Word2Vec-based network but without BERT model. In addition, we build DeepCoref *-LSI*, which is a variant of DeepCoref by replacing the BERT with LSI to produce context vectors, and other parts remain unchanged. DeepCoref-LSI is to demonstrate the advantage of fine-tuning BERT over IR-based technique for computing context representation. In this experiment, we only replace each component, and keep all hyper-parameter settings the same as RQ1.

RQ4 (Data sensitivity): RQ4 is to question the influence of data size on the performance of coreference classification. We conduct an experiment by increasingly enlarging the size of training data to examine the sensitivity between performance enhancement and data augmentation. We present the time consumption with the increase of data as well. We first randomly split all data into two parts (90% as training set, 10% as testing set), and keep the testing set unchanged across all experiments. Then, we enlarge the size of training set from 5% to 90%, where we use all training data when the ratio reaches 90%. For each ratio, we repeat the experiment for five times, and use a boxplot to show the evaluation metrics, and take the average time of five experiments as consuming time.

The experimental environment is a desktop computer equipped with a NVIDIA 1060 GPU, Intel Core i7 CPU, 16GB RAM, running on Ubuntu OS.

4.5 Evaluation metrics

We use commonly used metrics such as precision, recall and F1-Score [66], to evaluate the performance of coreference classification. We mentioned that we collect and annotate data in cooperation with CMB (see Sect. 4.2). Given the ground-truth label and predicted label from DeepCoref, we calculate metrics for each class and take their unweighted mean as final results. 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. In addition, some baseline methods need to measure the similarity to decide whether two entities are coreferent, so we use cosine similarity [24] to compute the distance between two vector representations.

Precision refers to the ratio of the number of correct predictions of positive labels to the total number of predictions of positive labels.

Recall refers to the ratio of the number of correct predictions of positive labels to the total number of positive labels.

F1-score is the harmonic mean of precision and recall.

Cosine similarity computes similarity as the normalized dot product of X and Y:

$$cosine(X, Y) = \frac{X \cdot Y}{\|X\| \times \|Y\|} \tag{4}$$

where X and Y are two vectors.

As for RQ2, given the ground-truth clusters, many metrics have been proposed in the literature to evaluate the clustering performance. Following the previous work [29], we select the metrics including Adjusted Rand Index (ARI) [38], Normalized Mutual Information (NMI) [61], homogeneity (HOM) [68, 69], completeness (COM) [68], V-measure (V-M) [68], and Fowlkes-Mallows Index (FMI) [50]. Higher value indicates better clustering performance for all six metrics. For clarity, we take all entities in our dataset as a fixed list, and we denote T as the ground-truth cluster labels, and P as the predicted cluster labels.

Adjusted rand index (ARI) takes values in $[-1, 1]$, reflecting the degree of overlap between the two clusters. It is improved based on Rand Index (RI). The raw RI is computed by $RI = \frac{a+b}{\binom{n}{2}}$, where a is the number of pairs that are assigned in the same cluster in T and also the same cluster in P, and b is the number of pairs that are assigned in different clusters both in T and P. $\binom{n}{2}$ is the total number of unordered pairs in a set of n entities. The raw RI score is then "adjusted for chance" into the ARI score using the following scheme:

$$ARI = \frac{RI - E(RI)}{max(RI) - E(RI)} \tag{5}$$

where $E(RI)$ is the expected value of RI. The ARI is thus ensured to have a value close to 0.0 for random labeling independently of the number of clusters and samples.

Normalized mutual information (NMI) is a normalization of the Mutual Information (MI) score to scale the results between 0 (no mutual information) and 1 (perfect correlation). It interpret the cluster performance information-theoretically.

$$NMI(T, P) = \frac{MI(T, P)}{\sqrt{H(T)H(P)}} \tag{6}$$

where $H(T)$ is the entropy of set T, i.e., $H(T) = -\sum_{i=1}^{|T|} p(i)log(p(i))$ and $p(i) = \frac{T_i}{N}$ is the probability that an object picked at random falls into class T_i. The $MI(T, P)$ is the mutual information between T and P: $MI(T, P) = \sum_{i=1}^{|T|} \sum_{j=1}^{|P|} p(i,j)log\left(\frac{p(i,j)}{p(i)p(j)}\right)$.

Homogeneity (HOM) measures all of its clusters contain *only* entities which are members of a single class by

$$h = 1 - \frac{H(T \mid P)}{H(T)} \tag{7}$$

Completeness (COM) measures *all* the entities that are members of a given class are elements of the same cluster by

$$c = 1 - \frac{H(P \mid T)}{H(P)} \tag{8}$$

where$H(T \mid P)$is the conditional entropy of the ground-truth classes given the predicted cluster labels, while $H(P \mid T)$ is calculated by swapping the positions of T and P.

V-measure (V-M) is the harmonic mean of HOM and COM.

$$v = 2 \times \frac{h \times c}{h + c} \tag{9}$$

Fowlkes-Mallows Index (FMI) ranges in [0, 1]. It is defined as the geometric mean between of the precision and recall:

$$FMI = \frac{TP}{\sqrt{(TP + FP) \times (TP + FN)}} \tag{10}$$

where TP is the number of True Positive (i.e., the number of pairs of entities that belong to the same clusters in both T and P), FP is the number of False Positive (i.e., the number of pairs of entities that belong to the same clusters in T but not in P) and FN is the number of False Negative (i.e., the number of pairs of entities that belong to the same clusters in P but not in T).

5 Results and analysis

5.1 Answering RQ1: coreference classification

This section is to demonstrate the effectiveness of DeepCoref comparing with baselines. Figure 5 presents the performance of coreference classification 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 6 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 average precision and recall of DeepCoref on projects reach 93.43% and 93.72%, Word2Vec 79.79% and 79.09%, LSI 81.91% and 81.90%, Levenstein 77.48% and 80.17%. The

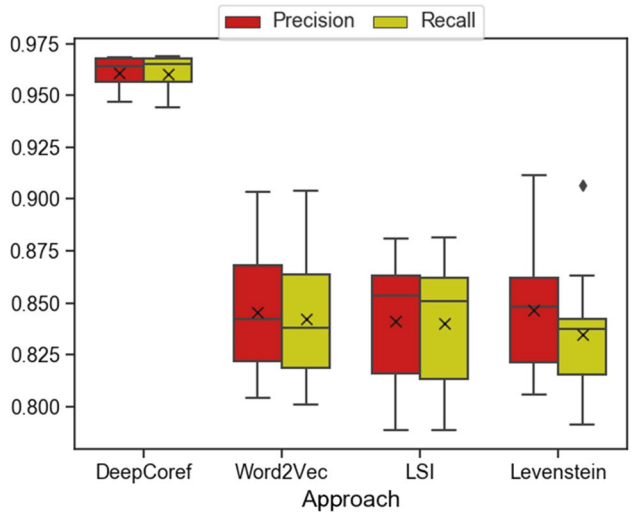


Fig. 5 RQ1: The performance of coreference classification of DeepCoref over baselines. The cross is the mean value of 10-fold cross-validation

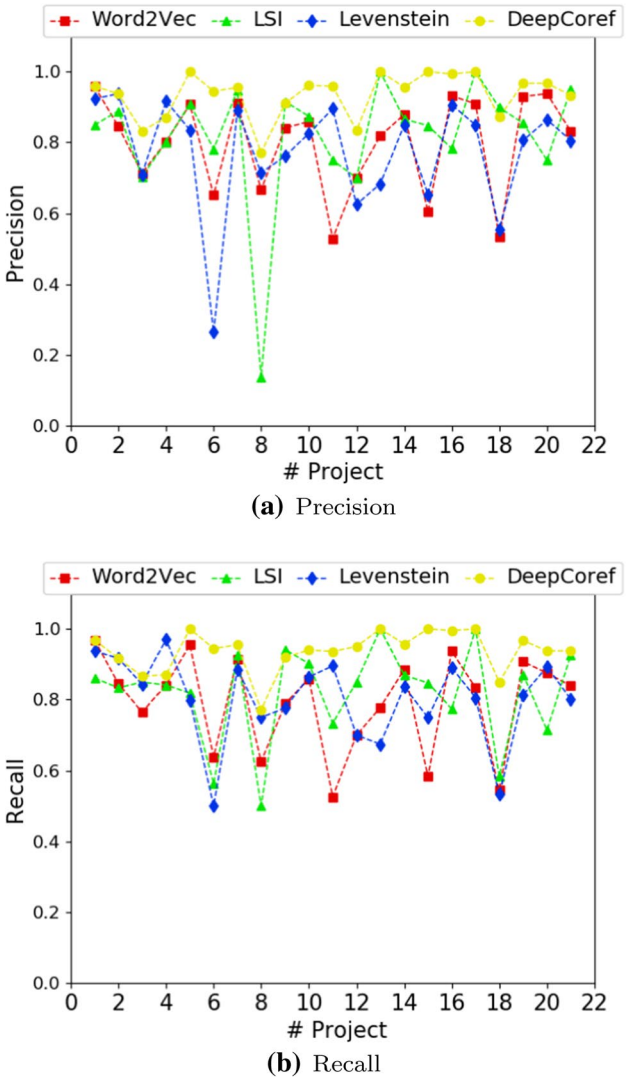


Fig. 6 RQ1: The performance of coreference classification of DeepCoref over baselines by project. The number of projects is 21

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 methods 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 contextual semantics. This is mainly because LSI is constructed based on statistical information on current training data, the representation ability 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 contextual semantics, thus more adaptable to different presentation styles.

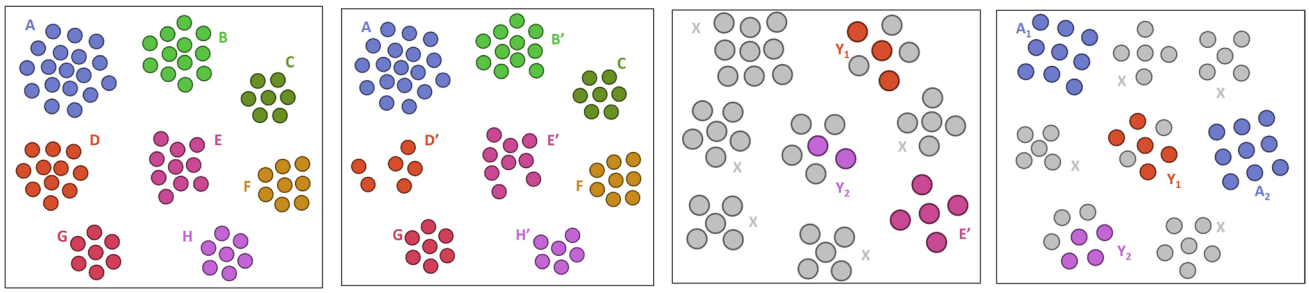
The reasons why DeepCoref noticeably outperforms the three baselines are:

- Type="SmallCaps">DeepCoref uses both sentence-level and word-level semantics thus can capture more information from contexts and entities.
- Type="SmallCaps">DeepCoref uses pre-trained models thus can benefit from large general corpora pre-training.
- Type="SmallCaps">DeepCoref uses the fine-tuning technique, which can improve adaptation on domain-specific tasks.

5.2 Answering RQ2: clustering and normalization

This section is to demonstrate the performance of clustering.

Clustering evaluation: Table 1 shows the clustering evaluation results on six metrics. Compared with the other three coreference techniques (i.e., Word2Vec, LSI and Levenstein), DeepCoref has the best performance on all six metrics. The comparison among them can help understand the impact of different coreference techniques on the clustering performance. For example, it is easy to understand that DeepCoref has the best performance of coreference classification, which leads to the best performance of clustering. The clustering performance of Word2Vec, LSI and Levenstein are different, especially on ARI and FMI, although they have little difference on the performance of coreference classification. This indicates that some misclassifications of entity pairs are fatal, which can generate new clusters or split the ground-truth clusters, and lead to performance decline of clustering. The best performance of clustering from DeepCoref verifies its effectiveness on coreference classification further. As for the comparison between DeepCoref and two clustering baselines, DeepCoref also outperforms K-Means



(a) Ground Truth **(b)** DeepCoref **(c)** K-Means **(d)** DBSCAN

Fig. 7 RQ2: The clustering results of ground truth, DeepCoref and baselines

Table 1 RQ2: Clustering Performance

Method	ARI	NMI	HOM	COM	V-M	FMI
DeepCoref	0.929	0.992	0.995	0.990	0.992	0.930
Word2Vec	0.794	0.973	0.980	0.967	0.973	0.796
LSI	0.835	0.978	0.963	0.986	0.978	0.840
Levenstein	0.582	0.958	0.927	0.985	0.958	0.631
K-Means	0.569	0.957	0.963	0.951	0.957	0.580
DBSCAN	0.680	0.966	0.968	0.964	0.966	0.684

and DBSCAN on all six metrics, which indicates our proposed clustering method is more effective than baselines.

The pairwise metrics (i.e., ARI and FMI) are sensitive to the cases when entity pairs which belong to the same cluster in the ground-truth are wrongly assigned into different clusters, or which belong to different clusters, are wrongly placed into the same cluster. We examine the clustering results and find that clusters obtained by baselines contain more newly formed clusters (e.g., *X* clusters in Fig. 7c) or splitted clusters (e.g., cluster A_1 and A_2 in Fig. 7d) compared with DeepCoref. Therefore, DeepCoref has an obvious advantage on ARI and FMI as shown in Table 1. The entropy-based metrics (i.e., NMI, HOM, COM and V-M) mainly evaluate the changes of two distributions based on information entropy theory. From the perspective of the number of clusters, there are totally 355 clusters in the ground-truth. The total number of clusters obtained by DeepCoref is 359, while the number of clusters obtained by K-Means and DBSCAN are 350 and 376, respectively. All methods don't change the distribution of the entire clusters very much, which conforms to the small improvement on NMI, HOM, COM and V-M in Table 1. However, the coreference detection task needs to assign coreferent (non-coreferent) entities into the same (different) clusters correctly, rather than only keeps the distributions of two clusters unchanged. Therefore, the pairwise metrics are more significant than entropy-based metrics to requirement engineers, which are exactly the advantages of DeepCoref over baselines.

Example analysis: More details can be found in Fig. 7, where we present top eight clusters of ground-truth, DeepCoref and two baselines. We can observe that the clustering result of DeepCoref is roughly the same as the ground-truth, but the clustering results of other baselines are somewhat different from the ground-truth. More specifically, we first compare Fig. 7a and b. The top eight clusters obtained by DeepCoref roughly conform to the ground-truth, where the cluster *A*, *C*, *F* and *G* are exactly the same, while the cluster B', D', E' and H' miss some entities. From the comparison between Fig. 7a and c, we can find that only the cluster E' is relatively accurate, which misses five entities. The cluster Y_1 and Y_2 consist of nodes from ground-truth cluster *D* and *H*, and other small (grey) clusters. The other clusters denoted as *X* are newly formed from entities which originally belong to small (grey) clusters in the ground-truth. The top eight clusters also changed compared with the ground-truth. As for Fig. 7(d), the cluster *A* in the ground-truth is splitted into two clusters (i.e., A_1 and A_2). The cluster Y_1 and Y_2 contain nodes from ground-truth cluster *D* and *H*, and other small (grey) clusters. Four newly formed clusters become top eight clusters.

We present the examples of coreferent/non-coreferent entities shown in Fig. 7. Table 2 demonstrates all 18 coreferent entities in cluster A, which are all correctly identified by DeepCoref. We observe that all 18 examples share some similar words in entities and related contexts, e.g., "warning", "list", and "stage" appearing in entities; "manager",

Table 2 RQ2: Examples of coreferent entities from cluster A of DeepCoref

#	Entity	Context
1	Warning customer list in tracking stage	Managers can view the *warning customer list in tracking stage* to check the completion of tracking stage in time.
2	List of alert customers in implementation stage	Managers can view the *list of alert customers in implementation stage* to check the completion of the implementation stage in time
3	Alert customer list in listening stage	Managers can view *alert customer list in listening stage* and check the completion of listening in time.
4	Customer in tracking stage	The manager can see the completion of *customer in tracking stage*.
5	Customer in implementation stage	Managers can see *customer in implementation stage* to follow their completion.
6	Customer in listening phrase	Managers can view *customer in listening phrase* to follow their completion.
7	Warning list during tracking stage	Account managers can examine the *warning list during tracking stage* to check the completion.
8	Alert list in implementation stage	Account managers can view the *alert list in implementation stage* to check their completion.
9	Warning list in demo stage	The managers can view the *warning list in demo stage* to examine the completion of the demo stage.
10	Warning customer information in tracking stage	Managers can check *warning customer information in tracking stage*, so that they can follow the completion.
11	Warning customer information during implementation	Managers can view *warning customer information during implementation* to track the completion of implementation stage.
12	Early-warning customer information in listening stage	Managers can check *early-warning customer information in listening stage*, so that they can follow the completion.
13	Warning user list in tracking stage	Managers can view *warning user list in tracking stage* to follow the completion of tracking stage.
14	Alert user in implementation stage	The managers can view *alert user in implementation stage* to follow the completion of implementation stage.
15	Warning customer in listening stage	Managers can view *warning customer in listening stage* to follow the completion of listening stage.
16	Warning customer table during tracking stage	Managers can view the *warning customer table during tracking stage* to check the completion of tracking stage.
17	List of warning customer in implementation stage	Managers can view the *list of warning customer in implementation stage* to check the completion of tracking stage in time.
18	Alarm list in demo stage	Managers can view the *alarm list in demo stage*, so that they can check the completion of demo stage.

Table 3 RQ2: Examples of non-coreferent entities from top 8 clusters of DeepCoref

cid	Normalized Entity	Context
A	Warning customer list in tracking stage	Managers can view the *warning customer list in tracking stage* to check the completion of tracking stage in time.
B'	Company announcement information	Account managers can check the *company announcement information* outside the bank to get more comprehensive information about the client.
C	Trading rules for the fund	Account managers can view the *trading rules for the fund* on the fund details page.
D'	Completion of monthly target	The center supervisor can view the performance statistics and the *completion of monthly target* of the account manager.
E'	Customer basic information	The system can provide a public interface to maintain the *customer basic information* and manage the customer maintenance information uniformly.
F	Project information	The project manager can modify the *project information* to keep it accurate.
G	Account information of cooperative institutions	The investment manager has the function of editing or deleting the *account information of cooperative institutions*, so that I can adjust the historical error information.
H'	Contract information	Project managers can add *contract information* and contracts can be reviewed paperless in the system

Table 4 RQ2: The performance improvement on the coreference classification brought by clustering

	Precision (%)	Recall (%)	F1 (%)
Classification	96.10	96.06	96.08
Clustering	98.52	98.53	98.53

"view", and "completion" appearing in contexts. Therefore, they can be easily inferred as coreferent entities by DeepCoref Then we turn to Table 3, it presents the normalized entity and related contexts of top 8 clusters. We can observe that both entities and contextual words are greatly different across clusters. The results of two tables shows the effectiveness of clustering coreferent entities, and the effectiveness of assigning a normalized name to each cluster.

Classification Improvement: Table 4 presents the performance improvement on coreference classification brought by clustering. We can observe that clustering can improve the precision, recall and F1 of coreference classification by 2.52%, 2.57% and 2.55%, respectively. We examine the differences in the results, and find that our proposed clustering method improves the classification performance by correcting the classification result with low confidence of two entities. Such misclassified cases tend to produce false nodes or miss true nodes between two clusters, which leads to wrongly or not merging clusters. Totally, our proposed clustering method corrects 87 samples which are misclassified by coreference network, although introduces 19 wrong corrections, which improves the overall performance of coreference classification.

In summary, the advantage of DeepCoref over other methods on evaluation metrics and network graphs indicates that it can cluster entities accurately based on the predicted results from context-wise coreference network. This is because compared with other methods, DeepCoref can capture semantics more efficiently by combining the semantics of both entities and contexts after performing coreference classification.

5.3 Answering RQ3: ablation experiment

This section is to demonstrate the different contribution of each component of DeepCoref to coreference classification. Figure 8 presents the performance of coreference classification 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

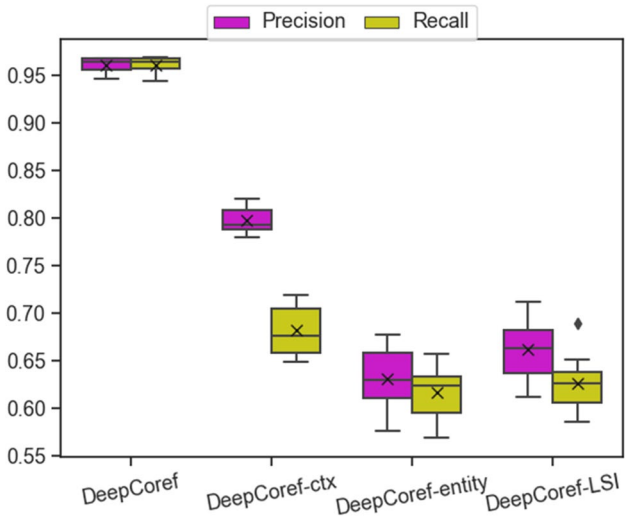


Fig. 8 RQ3: The performance of DeepCoref and its variants. The cross is the mean value of 10-fold cross-validation

from different components of context-wise coreference network. 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 contextual semantics are more effective than entity semantics. The improvement of precision and recall reaches 16.66% and 6.44% (differences between DeepCoref-ctx and DeepCoref-entity), 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% (differences between DeepCoref and DeepCoref-LSI), respectively.

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

5.4 Answering RQ4: data sensitivity

This section is to demonstrate the influence of different data size on coreference classification. Figure. 9 represents the relationship between performance and time consumption of DeepCoref when enlarging the size of the dataset. When the training set 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. We obtain the best performance at the last point where we use 90% data as training set. The inflection point

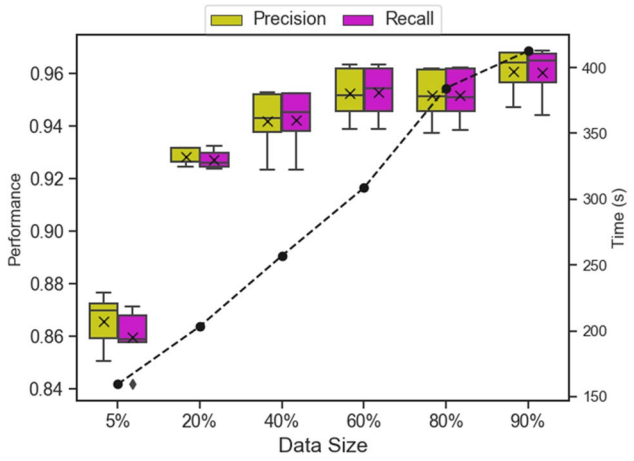


Fig. 9 RQ4: The performance of DeepCoref by data augmentation. The dotted line is time consuming

6 Discussion

6.1 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. We present the box-plot changing curve (each box includes 10 results from 10-fold cross-validation)

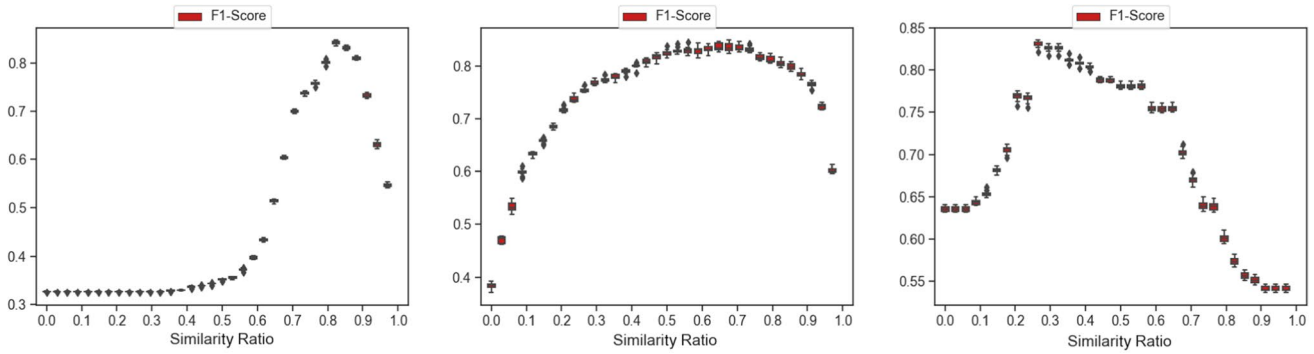


Fig. 10 The boxplot changing curve of F1-Score with the similarity ratio increasing from 0 to 1 by step 0.03 for each method. Each box contains results of one 10-fold cross-validation

occurs with 95.25% precision and 95.30% recall, when we use 60% data as training set. This indicates that DeepCoref is not very sensitive after the training set is greater than 60% (i.e., around 1100 in our experimental settings). Moreover, we can find that just using 20% data for training, the performance is also greater than 92% on average. It demonstrates that DeepCoref can address the low-resource problem well. In addition, the time consumption increases approximatively linearly from 159.42s to 412.52s. Considering our experimental environment, the time consumption of training the model is acceptable.

In summary, benefiting from large corpora pre-training and the fine-tuning technique, DeepCoref can reach a promising performance on a relatively small dataset. This alleviate the domain adaption problem such as low-resource problem to some extent.

of F1-Score for each method, 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 method. The final similarity ratio of each method is computed by an average of 10 optimal values.

Figure 10 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 methods significantly, and the optimal values are distinct for each method. Generally with the increase of similarity ratio, the F1-Score first rises and then declines for all methods. Nevertheless, this general trend exhibits a slight difference among these methods. 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

Table 5 The optimal similarity ratios of each method for each round of 10-fold cross-validation

#Fold	Word2Vec	LSI	Levenstein
#1	0.85	0.69	0.25
#2	0.85	0.68	0.25
#3	0.85	0.71	0.25
#4	0.85	0.68	0.25
#5	0.85	0.67	0.25
#6	0.85	0.59	0.25
#7	0.85	0.70	0.25
#8	0.85	0.59	0.25
#9	0.85	0.67	0.25
#10	0.85	0.67	0.25
Average	0.85	0.67	0.25

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.

Table 5 shows the optimal similarity ratios of each method for each round of 10-fold cross-validation. For each fold, 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 Fig. 10. 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, respectively. Hence, one should carefully tune the similarity ratios for each method, in order to achieve the best performance for a fair comparison.

6.2 Applicability

Resolving EC is usually a downstream task of entity extraction task in the pipeline of requirement analysis. In practice, entity extraction can rely on automated tools, and does not need much manual effort. The application scenario of DeepCoref is to resolve coreference on entities produced by these automated tools. There are several techniques for entity extraction such as general NLP tools [1, 22, 31] and domain-specific tools [42, 70]. When applying DeepCoref for detecting EC, one can firstly extract entities using tools mentioned above, then truncate contexts (see Sect. 3.1) and format the data to train the context-wise coreference network for coreference classification (see Sect. 3.2). You can also perform clustering and normalization to establish a non-coreferent entity dictionary. (see Sect. 3.3). Our method takes entities and related natural-language contextual text as input, and outputs coreferent clusters, and a normalized name for each coreferent cluster. We additionally list some key points when applying our method:

- 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.
- Our data are from financial domain. One should annotate about one thousand samples for fine-tuning the whole model to tackle domain adaption.
- The entities in our data are ready-made. If someone wants to apply our method but has no entities, he/she can select an automated tool to conduct entity extraction firstly.
- When applying to other languages, BERT and word embeddings must be pre-trained on corpus of corresponding languages.

7 Threats to validity

External validity: The external threats are related to the generalizability of the method. The experimental data are collected from the industry community, labeled manually, and evaluated on finance domain. However, we have retrieved as many projects as possible, and most annotators are domain practitioners and experts. The evaluation results by projects show that our method is generalized across projects, which alleviates the threat to some extent. In addition, our method uses BERT pre-trained on large general corpus and the fine-tuning technique, which alleviates the low-resource and generalizability problem. The Word2Vec we used is trained on Wikipedia dump, which might yield different results when training it on software requirement documents.

Internal validity: The internal threats relate to experimental errors and biases. Threats to internal validity may come from the entity extraction. The entities in our data are ready-made and well-maintained by our industry partners, which has a slight impact on our results. Additionally, the quality of extracted entities has little impact on this task. Because the idea of our proposed method is to learn whether two entities are coreferent based on the similarity of the combined semantics of entities and corresponding contexts. While low-quality entities will not impact our method to perform the semantic comparison, it will produce potentially semantically related but not properly extracted entities. Therefore, manual review on extracted entities is inevitable if automated entity extraction tools cannot work well in practice.

Construct validity: The construct threats relate to the suitability of evaluation metrics. We utilize precision and recall for coreference classification 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 Sect. 6.1). In addition, both predicted 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. As for clustering evaluation, we utilize six metrics including pairwise metrics and entropy-based metrics. These metrics measure clustering performance from different angles, which alleviates the threat.

8 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.

8.1 Detection of inconsistency

The amount of research on inconsistency detection has increased significantly in the past years. Mezghani et al. [56] used unsupervised machine learning algorithm, K-Means, for a redundancy and inconsistency detection in the RE context. They introduced a filtering method 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 [5] is a pattern-based unsupervised requirements clustering framework (based on K-Means algorithm), which makes use of machine-learning methods for requirements validation. The method 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. [33] 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. [65] introduced a novel method 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. [21] presented an NLP method to identify ambiguous terms between different domains and rank them by ambiguity score. The method 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 [6, 18, 19, 23, 67, 72]. Other works use heuristics to tackle coordination ambiguities (i.e., ambiguities brought by "and" or "or" conjunctions) [7] and anaphoric ones (i.e., ambiguities brought by pronouns) [78].

Our work complements to the existing researches in two aspects:

- It is a method to resolving EC in RE. Detecting EC can improve the readability and understandability of requirements.
- It is a deep learning method, which is more powerful and generic.

8.2 Coreference resolution

Our work is inspired by CDCR, so we review representative works on CR in recent years. For WDCR, Lee et al. [41] 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. [32] 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. [40] 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 [40], Barhom et al. [2] 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.

This work draws on the ideas of CDCR entity methods, but at the same time takes into account the characteristics of EC in RE. We summarize three differences between EC in RE and EC in general NLP tasks below:

- Coreferent entities in RE usually occur among multi-word noun phrases, and most entities are technical terms and relatively independent. It is not necessary to detect pronoun coreferences or traverse all probable mentions, because its main aim is to reach a shared understanding on some basic concepts among multiple stakeholders.
- Coreferent entities in RE are scattered in various sections of the natural-language requirement, which implies that the EC detection in RE are more dependent on the contextual semantics.
- The EC in RE is domain-specific tasks, which implies that there are domain adaptation problems such as low-resource problem. However, general EC tasks can obtain support from large general corpora or public knowledge bases.

9 Conclusion and future work

This paper resolves *Entity Coreference* in requirement engineering. We propose a *DEEP* context-wise method for entity *COREF*erence detection, which we name DeepCoref. The first step is to truncate contexts around entities. Then, we construct a context-wise coreference network for coreference classification. It consists of a fine-tuning BERT model for context representation, a Word2Vec-based network for entity representation, and 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. Finally, we assign entities which are coreferent to one concept into one same cluster by clustering, and assign a normalized name to each coreferent cluster by normalization. We investigate the effectiveness of DeepCoref with 1853 samples on 21 projects from the industry community. The experimental results of coreference classification show that our method significantly outperforms three baselines with average precision and recall of 96.10% and 96.06%, respectively. The clustering performance of DeepCoref is higher on six metrics compared with two baselines. In order to demonstrate the performance enhancement from different components of context-wise coreference network, we compare the performance of DeepCoref and three variants as well.

Type="SmallCaps">DeepCoref works better, mainly benefiting from its novel design of context-wise coreference network. 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, we only need to annotate a small amount of data for fine-tuning, which obtains a promising result, because DeepCoref can benefit from fine-tuning technique and pre-trained models (i.e., BERT and word embeddings) trained on large general corpora. It alleviates the problem of insufficient annotated resource and the high cost of manual annotation as well. The results also confirm that our method could effectively detect entity coreference from natural-language requirements, thus can facilitate reaching a shared understanding on entities among multiple stakeholders from different domains in an automated way.

In the future, we plan to add some event features into the entity representations based on what we have proposed in this work, because we observe that one entity is usually associated with a chain of events, such as CRUD (i.e., create, read, update and delete). Therefore, event information has the potential to help distinguish entities more precisely.

Acknowledgements This work is supported by the National Key Research and Development Program of China under grant No. 2018YFB1403400, 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. Arora C, Sabetzadeh M, Briand LC, Zimmer F (2017) Automated extraction and clustering of requirements glossary terms. IEEE Trans Software Eng 43(10):918–945
2. Barhom S, Shwartz V, Eirew A, Bugert M, Reimers N, Dagan I (2019) Revisiting joint modeling of cross-document entity and event coreference resolution. In: A. Korhonen, D.R. Traum, L. Màrquez (eds.) Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers, pp. 4179–4189. Association for Computational Linguistics
3. Baziotis C, Pelekis N, Doulkeridis C (2017) Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In: proceedings of the 11th international workshop on semantic evaluation (SemEval-2017), pp. 747–754. Association for Computational Linguistics, Vancouver, Canada
4. Beheshti S, Benatallah B, Venugopal S, Ryu SH, Motahari-Nezhad HR, Wang W (2017) A systematic review and comparative analysis of cross-document coreference resolution methods and tools. Computing 99(4):313–349
5. Belsis P, Koutoumanos A, Sgouropoulou C (2014) PBURC: a patterns-based, unsupervised requirements clustering framework for distributed agile software development. Requir Eng 19(2):213–225
6. Berry DM, Kamsties E (2005) The syntactically dangerous all and plural in specifications. IEEE Software 22(1):55–57
7. Chantree F, Nuseibeh B, De Roeck A, Willis A(2006) Identifying nocuous ambiguities in natural language requirements. In: 14th IEEE international requirements engineering conference (RE'06)
8. Chen X, Liu Z, Sun M (2014) A unified model for word sense representation and disambiguation. In: A. Moschitti, B. Pang, W. Daelemans (eds.) 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, pp. 1025–1035. ACL
9. Cleland-Huang J Mining domain knowledge [requirements]. IEEE Software 32(3): 16–19
10. Cohen WW, Ravikumar P, Fienberg SE (2003)A comparison of string distance metrics for name-matching tasks. In: S. Kambhampati, C.A. Knoblock (eds.) Proceedings of IJCAI-03 Workshop

on Information Integration on the Web (IIWeb-03), August 9-10, 2003, Acapulco, Mexico, pp. 73–78
11. Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa PP (2011) Natural language processing (almost) from scratch. J Mach Learn Res 12:2493–2537
12. Deerwester SC, Dumais ST, Landauer TK, Furnas GW, Harshman RA (1990) Indexing by latent semantic analysis. JASIS 41(6):391–407
13. Devlin J, Chang M, Lee K, Toutanova K (2019) 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 , pp. 4171–4186 (Long and Short Papers)
14. Doddington GR, Mitchell A, Przybocki MA, Ramshaw LA, Strassel SM, Weischedel RM (2004) 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
15. Ester M, Kriegel H, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: proceedings of the second international conference on knowledge discovery and data mining (KDD-96), Portland, Oregon, USA, pp. 226–231. AAAI Press
16. Fabbrini F, Fusani M, Gnesi S, Lami G (2001) The linguistic approach to the natural language requirements quality: Benefits of the use of an automatic tool. In: proceedings 26th annual NASA Goddard software engineering workshop, 2001
17. Falessi D, Cantone G, Canfora G (2013) Empirical principles and an industrial case study in retrieving equivalent requirements via natural language processing techniques. IEEE Trans Software Eng 39(1):18–44
18. Femmer H, Fernández DM, Wagner S, Eder S (2017) Rapid quality assurance with requirements smells. J Syst Softw 123:190–213
19. Femmer H, Kucera J, Vetro A (2014) On the impact of passive voice requirements on domain modelling. In: M. Morisio, T. Dybå, M. Torchiano (eds.) 2014 ACM-IEEE international symposium on empirical software engineering and measurement, ESEM '14, Torino, Italy, September 18-19, 2014, pp. 21:1–21:4. ACM
20. ...Fernández DM, Wagner S, Kalinowski M, Felderer M, Mafra P, Vetro A, Conte T, Christiansson M, Greer D, Lassenius C, Männistö T, Nayabi M, Oivo M, Penzenstadler B, Pfahl D, Prikladnicki R, Ruhe G, Schekelmann A, Sen S, Spínola RO, Tuzcu A, de la Vara JL, Wieringa RJ (2017) Naming the pain in requirements engineering - contemporary problems, causes, and effects in practice. Empirical Softw Eng 22(5):2298–2338
21. Ferrari A, Esuli A (2019) An NLP approach for cross-domain ambiguity detection in requirements engineering. Autom Softw Eng 26(3):559–598
22. Gemkow T, Conzelmann M, Hartig K, Vogelsang A (2018) 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, pp. 412–417
23. Gleich B, Creighton O, Kof L (2010) Ambiguity detection: Towards a tool explaining ambiguity sources. In: R.J. Wieringa, A. Persson (eds.) Requirements Engineering: Foundation for Software Quality, 16th International Working Conference, REFSQ 2010, Essen, Germany, June 30 - July 2, 2010. Proceedings, *Lecture Notes in Computer Science*, vol. 6182, pp. 218–232. Springer
24. Gomaa W, Fahmy AA (2013) A survey of text similarity approaches. Int J Comput Appl 68(13)
25. Harris ZS (1954) Distributional structure. WORD 10(2–3):146–162
26. Hey T, Keim J, Koziolek A, Tichy WF (2020) Norbert: Transfer learning for requirements classification. In: 28th IEEE International Requirements Engineering Conference, RE 2020, Zurich, Switzerland, August 31 - September 4, 2020, pp. 169–179. IEEE
27. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
28. Huang EH, Socher R, Manning CD, Ng AY(2012) 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, pp. 873–882 (2012)
29. Huang Y, Chen C, Xing Z, Lin T, Liu Y (2018) Tell them apart: distilling technology differences from crowd-scale comparison discussions. In: M. Huchard, C. Kästner, G. Fraser (eds.) Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE 2018, Montpellier, France, September 3-7, 2018, pp. 214–224. ACM
30. Hull MEC, Jackson K, Dick J (2002) Requirements engineering. Springer, London
31. Johann T, Stanik C, B, AMA, Maalej W (2017)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, pp. 21–30
32. Joshi M, Levy O, Zettlemoyer L, Weld DS (2019) BERT for coreference resolution: Baselines and analysis. In: K. Inui, J. Jiang, V. Ng, X. Wan (eds.) 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, pp. 5802–5807. Association for Computational Linguistics
33. Jürgens E, Deissenboeck F, Feilkas M, Hummel B, Schätz B, Wagner S, Domann C, Streit J (2010) Can clone detection support quality assessments of requirements specifications? In: J. Kramer, J. Bishop, P.T. Devanbu, S. Uchitel (eds.) Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2, ICSE 2010, Cape Town, South Africa, 1-8 May 2010, pp. 79–88. ACM
34. Khlif W, Haoues M, Sellami A, Ben-Abdallah H (2017) Analyzing functional changes in BPMN models using COSMIC. In: J.S. Cardoso, L.A. Maciaszek, M. van Sinderen, E. Cabello (eds.) Proceedings of the 12th International Conference on Software Technologies, ICSOFT 2017, Madrid, Spain, July 24-26, 2017, pp. 265–274. SciTePress
35. Khlif W, Sellami A, Haoues M, Ben-Abdallah, H (2018) Using COSMIC FSM method to analyze the impact of functional changes in business process models. In: E. Damiani, G. Spanoudakis, L.A. Maciaszek (eds.) Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering, ENASE 2018, Funchal, Madeira, Portugal, March 23-24, 2018, pp. 124–136. SciTePress
36. Kingma DP, Ba J (2015) 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
37. Kohavi R (1995) 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 2:1137–1145 (August 20-25 1995)
38. Lawrence, HubertPhipps, Arabie: comparing partitions. J Classif (1985)

39. Le QV, Mikolov T (2014) Distributed representations of sentences and documents. In: Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014. JMLR Workshop and Conference Proceedings 32:1188–1196
40. Lee H, Recasens M, Chang AX, Surdeanu M, Jurafsky D (2012) Joint entity and event coreference resolution across documents. In: J. Tsujii, J. Henderson, M. Pasca (eds.) 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, pp. 489–500. ACL
41. Lee K, He L, Lewis M, Zettlemoyer L (2017) End-to-end neural coreference resolution. In: M. Palmer, R. Hwa, S. Riedel (eds.) Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017, pp. 188–197. Association for Computational Linguistics
42. Li M, Yang Y, Shi L, Wang Q, Hu J, Peng X, Liao W, Pi G (2020) Automated extraction of requirement entities by leveraging LSTM-CRF and transfer learning. In: IEEE International Conference on Software Maintenance and Evolution, ICSME 2020, Adelaide, Australia, September 28 - October 2, 2020, pp. 208–219. IEEE
43. Li M, Zhang T, Chen Y, Smola AJ (2014) Efficient mini-batch training for stochastic optimization. In: S.A. Macskassy, C. Perlich, J. Leskovec, W. Wang, R. Ghani (eds.) The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014, pp. 661–670. ACM
44. Li S, Zhao Z,Hu R, Li, W, Liu T, Du X (2018) Analogical reasoning on chinese morphological and semantic relations. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers, pp. 138–143
45. Lian X, Rahimi M, Cleland-Huang J, Zhang L, Ferrai R, Smith M (2016) Mining requirements knowledge from collections of domain documents. In: 24th IEEE International Requirements Engineering Conference, RE 2016, Beijing, China, September 12-16, 2016, pp. 156–165. IEEE Computer Society
46. Logeswaran L, Chang M, Lee K, Toutanova K, Devlin J, Lee H (2019) Zero-shot entity linking by reading entity descriptions. In: A. Korhonen, D.R. Traum, L. Màrquez (eds.) Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers, pp. 3449–3460. Association for Computational Linguistics
47. Lu J, Ng V (2017) Joint learning for event coreference resolution. In: R. Barzilay, M. Kan (eds.) Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers, pp. 90–101. Association for Computational Linguistics
48. Mahmoud A, Niu N (2015) On the role of semantics in automated requirements tracing. Requir Eng 20(3):281–300
49. Maletic JI, Marcus A (2000) 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, pp. 46–53
50. Mallows EBFL (1983) A method for comparing two hierarchical clusterings. J Am Stat Assoc 78(383):553–569
51. Manning CD, Raghavan P, Schütze H (2010)Introduction to information retrieval
52. Marcus A, Maletic JI (2001) 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, pp. 107–114
53. Marcus A, Maletic JI (2003) 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, pp. 125–137
54. Melamud O, Goldberger J, Dagan I(2016) 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, pp. 51–61
55. Melamud O, McClosky D, Patwardhan S, Bansal M (2016) The role of context types and dimensionality in learning word embeddings. In: K. Knight, A. Nenkova, O. Rambow (eds.) 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, pp. 1030–1040. The Association for Computational Linguistics
56. Mezghani M, Kang J, Sèdes F(2018) Industrial requirements classification for redundancy and inconsistency detection in SEMIOS. In: G. Ruhe, W. Maalej, D. Amyot (eds.) 26th IEEE International Requirements Engineering Conference, RE 2018, Banff, AB, Canada, August 20-24, 2018, pp. 297–303. IEEE Computer Society
57. Mihalcea R, Tarau P(2004) Textrank: Bringing order into text. Emnlp pp. 404–411
58. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. In: Y. Bengio, Y. LeCun (eds.) 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings
59. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) 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, pp. 3111–3119
60. Misra J, Das S (2013) Entity disambiguation in natural language text requirements. In: P. Muenchaisri, G. Rothermel (eds.) 20th Asia-Pacific Software Engineering Conference, APSEC 2013, Ratchathewi, Bangkok, Thailand, December 2-5, 2013 - Volume 1, pp. 239–246. IEEE Computer Society
61. Nguyen XV, Epps J, Bailey J (2010) Information theoretic measures for clusterings comparison: variants, properties, normalization and correction for chance. J Mach Learn Res 11:2837–2854
62. Palangi H, Deng L, Shen Y, Gao J, He X, Chen J, Song X, Ward RK(2015) Deep sentence embedding using the long short term memory network: Analysis and application to information retrieval. CoRR arXiv: abs/1502.06922
63. Pennington J, Socher R, Manning CD (2014) Glove: Global vectors for word representation. In: A. Moschitti, B. Pang, W. Daelemans (eds.) 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, pp. 1532–1543. ACL
64. Radford A, Narasimhan K, Salimans T, Sutskever I (2018) Improving language understanding by generative pre-training
65. Rago A, Marcos CA, Diaz-Pace JA (2016) Identifying duplicate functionality in textual use cases by aligning semantic actions. Softw Syst Model 15(2):579–603
66. Rijsbergen C (1979) Information retrieval (2nd edition)
67. Rosadini B, Ferrari A, Gori G, Fantechi A, Gnesi S, Trotta I, Bacherini S (2017) Using NLP to detect requirements defects: An industrial experience in the railway domain. In: P. Grünbacher, A. Perini (eds.) Requirements Engineering: Foundation

for Software Quality - 23rd International Working Conference, REFSQ 2017, Essen, Germany, February 27 - March 2, 2017, Proceedings,. Lecture Notes in Computer Science 10153:344–360 (**Springer**)
68. Rosenberg A, Hirschberg J (2007) V-measure: A conditional entropy-based external cluster evaluation measure. In: J. Eisner (ed.) EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic, pp. 410–420. ACL
69. Sato-Ilic M (2000) On evaluation of clustering using homogeneity analysis. In: Proceedings of the IEEE International Conference on Systems, Man & Cybernetics: "Cybernetics Evolving to Systems, Humans, Organizations, and their Complex Interactions", Sheraton Music City Hotel, Nashville, Tennessee, USA, 8-11 October 2000, pp. 3588–3593. IEEE
70. Shi L, Li M, Xing M, Wang Y, Wang Q, Peng X, Liao W, Pi G, Wang H (2020) Learning to extract transaction function from requirements: an industrial case on financial software. In: ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020, pp. 1444–1454. ACM
71. Srivastava N, Hinton GE, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
72. Tjong SF, Berry D M (2013) The design of SREE - A prototype potential ambiguity finder for requirements specifications and lessons learned. In: J. Dörr, A.L. Opdahl (eds.) Requirements Engineering: Foundation for Software Quality - 19th International Working Conference, REFSQ 2013, Essen, Germany, April 8-11, 2013. Proceedings. Lecture Notes in Computer Science 7830:80–95 (**Springer**)
73. Turian JP, Ratinov L, Bengio Y (2010) Word representations: A simple and general method for semi-supervised learning. In: J. Hajic, S. Carberry, S. Clark (eds.) ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden, pp. 384–394. The Association for Computer Linguistics
74. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. In: Guyon I, von Luxburg U, Bengio S, Wallach HM, Fergus R, Vishwanathan SVN, Garnett R (eds) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4–9 December 2017. Long Beach, CA, USA, pp 5998–6008
75. Wang W, Niu N, Liu H, Niu Z (2018) Enhancing automated requirements traceability by resolving polysemy. In: 26th IEEE International Requirements Engineering Conference, RE 2018, Banff, AB, Canada, August 20-24, 2018, pp. 40–51
76. Wang Y, Shi L, Li M, Wang Q, Yang Y (2020) A deep context-wise method for coreference detection in natural language requirements. In: 28th IEEE International Requirements Engineering Conference, RE 2020, Zurich, Switzerland, August 31 - September 4, 2020, pp. 180–191. IEEE
77. Wolf T, Debut L, Sanh V, Chaumond J, Delangue C, Moi A, Cistac P, Rault T, Louf R, Funtowicz M, Davison J, Shleifer S, von Platen P, Ma C, Jernite Y, Plu J, Xu C, Scao TL, Gugger S, Drame M, Lhoest Q, Rush AM (2020) Transformers: State-of-the-art natural language processing. In: proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations, pp. 38–45. Association for Computational Linguistics, Online
78. Yang H, De Roeck A, Gervasi V, Willis A, Nuseibeh B (2011) Analysing anaphoric ambiguity in natural language requirements. Requir Eng 16(3):163–189