Automated Extraction of Requirement Entities by Leveraging LSTM-CRF and Transfer Learning

Mingyang Li*‡, Ye Yang¶, Lin Shi *‡, Qing Wang*†‡[1], Jun Hu *‡,
Xinhua Peng§, Weimin Liao§ and Guizhen Pi§
*Laboratory for Internet Software Technologies, Institute of Software Chinese Academy of Sciences, Beijing, China
¶ Stevens Institute of Technology, Hoboken, NJ, USA
†State Key Laboratory of Computer Sciences, Institute of Software Chinese Academy of Sciences, Beijing, China
‡University of Chinese Academy of Sciences, Beijing, China
§China Merchants Bank, Shenzhen, China
Email: mingyang@itechs.iscas.ac.cn, yyang4@stevens.edu, {shilin, wq, hujun}@iscas.ac.cn
{joeyxhpeng, liaowm, p98119}@cmbchina.com

***Abstract*—Requirement entities, "explicit specification of concepts that define the primary function objects", play an important role in requirement analysis for software development and maintenance. It is a labor-intensive activity to extract requirement entities from textual requirements, which is typically done manually. A few existing studies propose automated methods to support key requirement concept extraction. However, they face two main challenges: lack of domain-specific natural language processing techniques and expensive labeling effort. To address the challenges, this study presents a novel approach named RENE, which employs LSTM-CRF model for requirement entity extraction and introduces the general knowledge to reduce the demands for labeled data. It consists of four phases: 1) Model construction, where RENE builds LSTM-CRF model and an isomorphic LSTM language model for transfer learning; 2) LSTM language model training, where RENE captures general knowledge and adapt to requirement context; 3) LSTM-CRF training, where RENE trains the LSTM-CRF model with the transferred layers; 4) Requirement entity extraction, where RENE applies the trained LSTM-CRF model to a new-coming requirement, and automatically extracts its requirement entities. RENE is evaluated using two methods: evaluation on historical dataset and user study. The evaluation on the historical dataset shows that RENE could achieve 79% precision, 81% recall, and 80% F1. The evaluation results from the user study also suggest that RENE could produce more accurate and comprehensive requirement entities, compared with those produced by engineers.**

***Index Terms*—Requirement Entity, Sequence Tagging, LSTM-CRF, Transfer Learning**

I. INTRODUCTION

Successful software products require continuous changes triggered by user requirements. If inappropriately managed, chaotic requirements may lead to ambiguous or duplicate features, lack of visibility in requirement dependency, poor scope and cost estimation of software projects. To address these issues, an effective mechanism is to establish and maintain a function dictionary which characterizes the key functional objects of the software system. In general, such functional objects can be represented as requirement entities, as "explicit specifications of real-world objects in the requirement context", and define the scope and functions to be implemented or enhanced in the software [1]–[4]. Additional benefits of requirement entities are reported to support new requirements identification [5], requirement change analysis [6], and effort estimation based on requirement changes [7]. However, in practice, the activity to extract requirement entities from textual requirements is generally done in a manual manner, due to lack of automated support. This labor-intensive manual process frequently leads to a redundant, missing, and chaotic list of requirement entities.

In essence, the problem of automated requirement term extraction is similar to the classical term extraction problem in the machine learning domain. There are a plethora of machine learning-based approaches to address the general term extraction problem [8]–[10], which could be potentially applicable to the software requirement domain. However, these approaches rely on a large volume of labeled resources to train a promising model. The labeling process is very expensive, let alone the ubiquitous difficulty in getting access to industrial requirements [11], [12]. Additionally, a few studies construct and apply heuristic linguistic rules, to parsed requirement corpus leveraging Natural Language Processing (NLP) techniques. Examples of such methods are part-of-speech patterns, grammar rule and text chunking [13]–[16]. Unfortunately, such NLP techniques are often trained on the general corpora which mainly contain a common vocabulary of terms, and errors might be produced when dealing with requirements with the obscure business terms due to the out-of-vocabulary problem, leading to poor performance under domain-specific context [17]. Therefore, there is a lack of studies on domain-specific NLP techniques.

In these existing studies, several different but related terms are named, such as general named entities [8], [9] and glossary terms [10], [13]–[16]. For example, general named entities refer to all general concept contained in a textual document [18], [19]. Glossary terms, defined as "salient terms in documents" [14], aim to help stakeholders get familiar with

[1]The corresponding author

2576-3148/20/$31.00 ©2020 IEEE
DOI 10.1109/ICSME46990.2020.00029

business knowledge the technical terminologies in a domain. It is observed that there is a lack of clear definition for these related terms, and existing definitions tend to be very inclusive, i.e. not only including domain-specific terms but also general terms loosely related to the requirement domain. When it comes to requirements-based software measurement and estimation, the inclusion of trivial, loosely relevant general terms could potentially lead to over-estimation [1], [2].

Therefore, in this study, we consider requirement entities which are context-sensitive, domain-specific, functionality-oriented terms, to support more effective requirement management and accurate estimation. Intuitively, this should be a subset of general named entities and glossary terms. To support effective requirement entity extraction, we propose a novel approach named RENE (**R**equirement **EN**tity **E**xtraction), for automatic requirement entity extraction. RENE employs LSTM-CRF model and transfer learning which takes requirement entity extraction as a sequence tagging task and introduces the general knowledge to reduce the demands for labeled data It consists of four phases: 1) Model construction, where RENE builds LSTM-CRF model and an isomorphic LSTM language model for transfer learning; 2) LSTM language model training to capture general knowledge and adapt to specific requirements; 3) LSTM-CRF training to train the LSTM-CRF model with the transferred layers; 4) Requirement entity extraction, where RENE applies the trained LSTM-CRF model to a new-coming requirement, and automatically extract its requirement entities. RENE is evaluated on 3,586 requirements from 20 on-going software systems in a financial company, compared with three state-of-art baselines. The evaluation results show that RENE could achieve 81% precision, 79% recall, and 80% F1 on average, and significantly outperform the baseline approaches. The evaluation results also indicate that training steps in RENE could contribute to the performance. The sample size sensitiveness evaluation shows that RENE could achieve promising performance even using very limited training data. Besides, the evaluation results from applying RENE to 11 ongoing projects show that RENE could achieve promising performance and outperform the engineer's manual extraction in practices.

The contributions of this paper are as follows: 1) The integration of LSTM-CRF and transfer learning to balance the dual learning needs of general knowledge as well as domain adaptation; 2) The construction and training of RENE to support automated extraction of requirement entities, overcoming existing challenges; 3) The evaluation on industrial requirement dataset from the on-going software projects, with promising results;

The remainder of the paper is organized as follows. Section II introduces the background. Section III elaborates the approach. Section IV presents the experiment design. Section V describes the results. Section VI discusses the benefits, learned lessons, and threats to validity. Section VII introduces the related work. Section VIII concludes our work.

II. BACKGROUND

A. Sequence Tagging and LSTM-CRF Model

Sequence tagging is a widely-used technique which is to predict the tag sequence given the input sequence [20] Let $X = (x_1, x_2, x_3, x_4, ..., x_n)$ be the input sequence where x_i is the element in the input sequence X. Sequence tagging task aims at finding the most optimal label sequence $Y = (y_1, y_2, y_3, y_4..., y_n)$ given X where y_i is the label given to corresponding x_i. Considering that the natural language is naturally sequence of words, sequence tagging has been popularly-used in many NLP tasks, such as Part-of-Speech tagging, text chunking, and named entity recognition, etc. [20], [21]. The sequence tagging task could be solved with many machine learning algorithms such as Maximum Entropy Model [22], Hidden Markov Model [23], and Conditional Random Fields (CRF) [20].

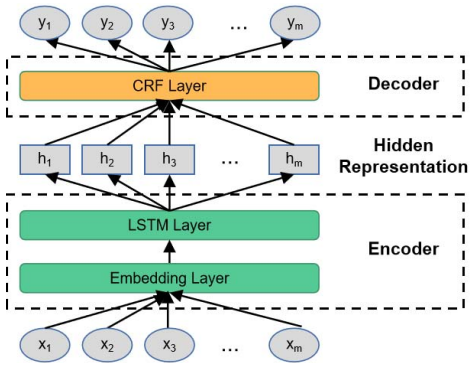

Fig. 1. The LSTM-CRF Model

LSTM-CRF model a kind of sequence tagging approach, which combines neural network and CRF algorithm. It is firstly proposed by Huang et al. [24], and has achieved the state-of-the-art results on many NLP tasks [25], [26]. The structure of the LSTM-CRF model is shown in Figure 1. There are three layers in the LSTM-CRF model, the first two layers, i.e. the embedding layer and the LSTM layer, responsible for encoding inputs into hidden representation, and the last CRF layer is to decode the hidden representation into label sequence. One of the advantages of the embedding layer and LSTM layer is the ability to encode sophisticated semantic and contextual information in the text. Thus, LSTM-CRF model typically achieves better performance than a single CRF model with hand-crafted features. However, LSTM-CRF model requires a large volume of labeled data for training, which limits its application to domains with the shortage of labeled data.

B. Language Model

Language model (LM) aims to capture regularities of natural language [27]. It is used to generate the probability distribution of various linguistic units, such as words, sentences, and whole documents [28]. Given the word sequence [x_1, x_2, ..., x_m], it

computes the probability of the sequence by modelling the probability of word x_k given the history (x_1, x_2, ..., x_{k-1}):

$$p\left(x_1, x_2, ..., x_m\right) = p(x_1) \prod_{k=1}^{m} \left(x_k | x_1, ..., x_{k-1}\right) \quad (1)$$

Language model could be trained using an unsupervised corpus. The trained language model could capture the linguistic properties such as grammatical structure, semantic and context information, and other knowledge.

C. Transfer Learning

Transfer learning is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem [29]. Given the source learning task T_s and target learning task T_t, the T_s shares the common or related knowledge with T_t. Transfer learning aims to help improve the performance of T_t with the help of the knowledge in T_s. For NLP tasks, supervised deep learning has become the most popular technique [30]. However, supervised deep learning typically requires a large number of labeled instances to fit numerous parameters in the neural network. Transfer learning could mitigate the situation. With the help of the general knowledge learned from T_s, only a few parameters in T_t need to be learned from small-sized datasets. To avoid the extra labeled cost, unsupervised tasks are usually set as the source tasks. In our study, we set the source task as the language model, and target task as the requirement entity extraction. We consider that the linguistic properties learned by the language model could boost the performance of the target requirement entity extraction task.

III. APPROACH

Figure 2 illustrates the overview of our approach, RENE. It consists of four phases: 1) Model construction, where RENE builds LSTM-CRF model and an isomorphic LSTM language model for transfer learning; 2) LSTM language model to capture general knowledge and adapt to specific requirements; 3) LSTM-CRF training to train the LSTM-CRF model with the transferred layers; 4) Requirement entity extraction, where RENE applies the trained LSTM-CRF model to a new-coming requirement, and automatically extract its requirement entities. Following introduces the details of the four phases.

A. LSTM Language Model Construction

RENE treats requirement entity extraction as a sequence tagging task and employs the LSTM-CRF model (the structure is shown in Figure 1) to encode the semantic and contextual information of requirement entities. In order to alleviate the data deficiency problem, RENE adopts an LM-based transfer learning strategy which sets the source task as the language model and constructs an isomorphic LSTM language model to capture the general knowledge for knowledge transfer.

There are three layers in the LSTM language model, i.e., an embedding layer, an LSTM layer, and a softmax layer. For each sentence, the LSTM language takes the corresponding word sequence as the input. The embedding layer is used to embed each word into a continuous space where the semantically similar words are placed close to each other, and LSTM layer is to encode the contextual information for each word. After that, the the LSTM layer returns a hidden representation [h_1, h_2, ..., h_m]. Then, the hidden representation is normalized into the probability distribution of the input sentence using the softmax function [31].

B. LSTM Language Model Training

In this phase, the LSTM language model is trained to capture the general knowledge and adapt to the specific requirements domain. Following introduces the two steps in the phrase.

1) Pre-training the LSTM language model with general corpus: This step is implemented by pre-training the LSTM language model with the easily-collected general corpus. Then, we build the training instances from the general corpus. The general corpus is pre-processed respectively following two steps: (1) splitting issue description text into individual sentences; and (2) applying standard data cleaning pipeline including special character removal, tokenization, and lowercase conversion to each sentence. After data pre-processing, each sentence is converted into a word sequence [w_1, w_2, ..., w_n], where w_i is the each word in the sentence, and n is the length of the sentence. Then, RENE models the distribution of each sentence. The input for the LSTM language model is set as [$< s >$, w_1, .., W_{m-1}], and the output is set as [w_1, w_2, .., w_m] where $< s >$ is a placeholder to align the input and output. The basic idea is to predict the words from left-to-right, to capture the statistical regularity within the general corpus. The pair ([$< s >$, w_1, .., w_{m-1}], [w_1, w_2, .., w_m]) is regarded as a training instance.

With the training instances, RENE pre-trains the LSTM language model in an iterative manner. During an iteration, all the instances are fed into the LSTM language model. For one time, an instance is sent into the LSTM language model. Through the embedding layer, the LSTM layer, and the softmax layer, the LSTM language model produces the probability sequence [$P(x_1)$, $P(x_2|x_1)$, ..., $P(x_m|x_1x_2...x_{m-1})$]. Then, the loss of the instance is defined as Cross-Entropy [32] in the following equation:

$$Loss_{lm} = \frac{1}{m} \sum_{j=1}^{m} -x_j \cdot \log P(x_j | x_1 x_2 ... x_{j-1}) \quad (2)$$

where x_j is the output for the current instance, and $P(x_j|x_1x_2...x_{j-1})$ is the probability of predicting x_j given the previous sequence [x_1, x_2, ..., x_{j-1}]. With the loss, each model parameter is updated using the following equation:

$$\theta_i := \theta_i - \eta \frac{\partial}{\partial \theta_i} Loss_{lm} \quad (3)$$

where θ_i is each model parameter in the embedding layer and the LSTM layer, the $\frac{\partial}{\partial \theta_i}$ is the operator of Gradient for the parameter θ_i, and η is the learning rate which could control the degree of parameter update. The iterative process terminates until $Loss_{lm}$ converges.

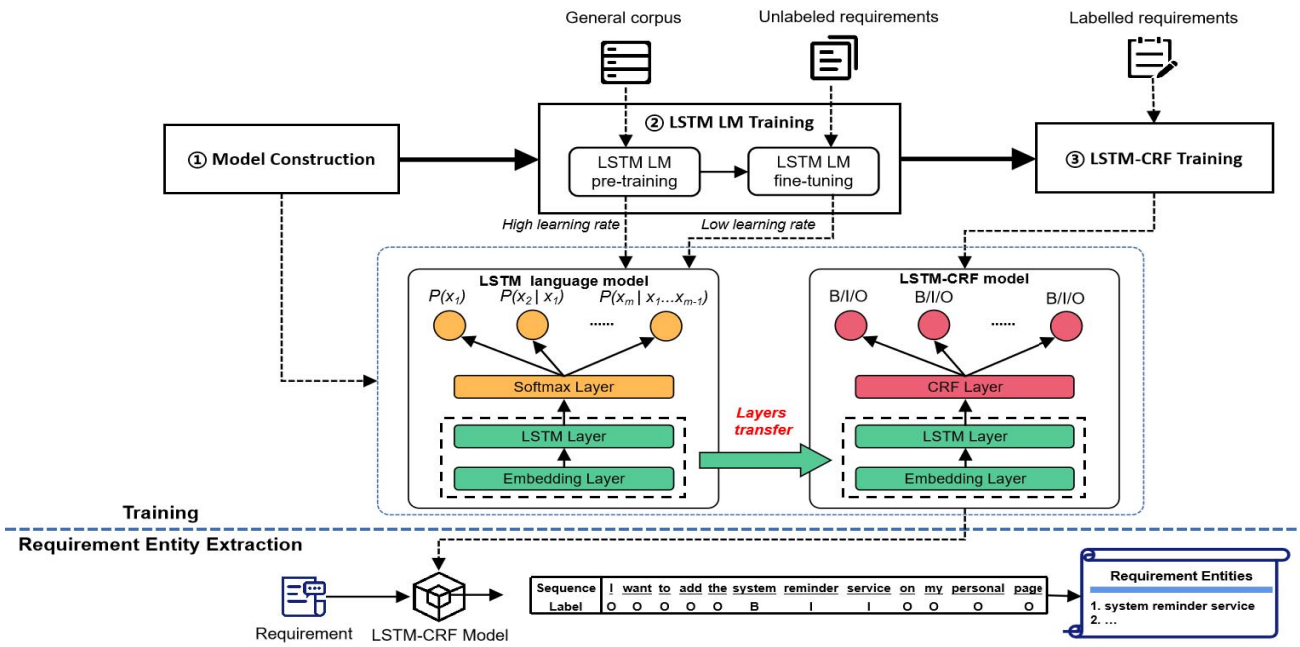

Fig. 2. The Approach Overview

2) Fine-tuning the LSTM language Model with unlabeled requirements: This step is to introduce a small amount of unlabeled data to fine-tune the LSTM language model to fill the linguistic gap between the general corpus and the specific domain. This step is similar to the previous pre-training step but uses unlabeled requirements as input instead. Specifically, RENE builds the LM instances from the unlabeled requirements, sends all the instances into the pre-trained LSTM language model. The training is also conducted iteratively. In each iteration, the loss for each instance is calculated by Cross-Entropy of the input word sequence and output probability sequence using the Equation 2, and model parameters are updated using the Equation 3. The difference is that RENE sets a smaller learning rate η in this step. The assumption is that, after pre-training the LSTM language model, the LSTM language model has captured the general knowledge from the general corpus. Requirements adaptation could be implemented with a smaller parameter update.

C. LSTM-CRF Model Training

First, RENE builds the sequence tagging instances from the labeled requirements. Given a requirement R and corresponding requirement entities $E = [RE_1, RE_2, ..., RE_n]$, all the textual contents in R are also split into sentences and processed by the standard data cleaning pipeline respectively. At the same time, RE_i is also processed by the data cleaning pipeline. After that, each sentence is converted into a word sequence $W_s = [w_1, w_2, ..., w_m]$, and each RE_i is converted into a word sequence W_{RE} which is the sub-sequence of W_s. For each W_{RE}, RENE determine its location in W_s by sub-sequence matching. Then, RENE gives a label y_i to each word w_i, which indicates the location of RE_i. The y_i in represented in the *BIO* format [33], [34]:

- *B-label (Beginning)*: The word is the beginning of the requirement entity.
- *I-label (Inside)*: The word is inside requirement entity but not the first within the requirement entity.
- *O-label (Outside)*: The word is outside the requirement entity.

Finally, each word sequence and corresponding label sequence is used as an instance.

After pre-training and fine-tuning the LSTM language model, RENE transfers the embedding layer and LSTM layer in the LSTM language model to the LSTM-CRF model, and only trains the CRF layer using the sequence tagging instance built from the labeled requirements. Similarly, this step is conducted iteratively. In each iteration, all the sequence tagging instances are sent into the LSTM-CRF network. For each time, LSTM-CRF network receives the input sequence $[w_1, w_2, ..., w_m]$. Through the embedding layer, the LSTM layer and the CRF layer, the network produces the predicted label sequence $[\hat{y_1}, \hat{y_2}, ..., \hat{y_m}]$, where $\hat{y_m}$ is one of three labels in the BIO format. The loss is defined as Cross Entropy:

$$Loss_{st} = \frac{1}{m}\sum_{j=1}^{m} -y_j \cdot \log p(\hat{y_j}) \tag{4}$$

where y_j is the ground truth label and $p(\hat{y_j})$ is the probability of predicting the label as y_j. The each parameters in the CRF layer is updated using the following equation:

$$\theta_i := \theta_i - \eta \frac{\partial}{\partial \theta_i} Loss_{st} \tag{5}$$

The iteration stops until the loss converges.

D. Requirement entity extraction

When a new requirement arrives, the trained LSTM-CRF receives its sentence representation [x_1, x_2, ..., x_m], and returns a label sequence [y_1, y_2, ..., y_m], where y_i (in the *BIO* format) is the label given to x_i. RENE forward scans the label sequence. Using the labels "B" and "I", RENE determines the location of each requirement entity in the word sequence. For example, given a sentence *"I want to add the system reminder service on my personal page"*, the word sequence and corresponding label sequence are shown in the requirement entity extraction phrase in Figure 2. The "Sequence" row corresponds to the word sequence of the sentence and the "Label" row represents the corresponding label sequence. In this example, the labels "B" and "I" correspond to the words "system", "reminder" and "service", and the next sequence label "O" represents the corresponding word no longer belongs to the current requirement entity. Therefore, the identified requirement entity will be "system reminder service" from the word sequence. Following this, all requirement entities could be extracted after parsing through the labeled sequence.

IV. EXPERIMENT DESIGN

A. Research Questions

The evaluation addresses the following four research questions:

RQ1: (Baseline Comparison) How does the performance of RENE compare with the state-of-art baselines? To evaluate the effectiveness of RENE, we evaluate its performance and compare it with three state-of-the-art baselines.

RQ2: (Transfer Learning Evaluation) To what extent does each training step contribute to RENE? There are mainly three training steps in RENE, i.e., pre-training the LSTM language mode, fine-tuning the LSTM language model, and train the LSTM-CRF model. We conduct experiments under different combinations of these training steps, to investigate to what extent each step contributes to the performance of RENE.

RQ3: (Sensitivity Analysis) To what extent is RENE sensitive to different sample sizes? We train RENE using different sizes of unlabeled requirements and labeled requirements, and evaluate the performances to investigate how many requirements are required to train an effective LSTM-CRF model.

RQ4: (Usefulness Analysis) How does RENE work in real-world application? We conduct a user study to investigate the usefulness of RENE in practices. we randomly sample the projects operating on the continuously-evolved business systems. The requirement entities extracted by RENE and engineers are manually reviewed by requirement experts respectively.

B. Subject Datasets

As illustrated in Fig 2, the subject datasets include two different corpora, i.e. general corpus extracted from Wikipedia for LSTM pre-training purpose, and domain-specific corpus from an industry collaborator for LSTM language model fine-tuning and LSTM-CRF model training purposes. Next, we will introduce these two datasets in detail.

TABLE I
SUMMARY OF LABELED AND UNLABELED REQUIREMENTS

system ID	business field	labeled requirement		unlabeled requirements
		requirements	requirement entities	requirements
1	Personal Loan	805	534	801
2	Merchant Management	108	146	124
3	Measurement Tool	88	124	72
4	Payment	68	68	60
5	Data Warehouse	64	88	92
6	Cloud Computing	192	208	188
7	Marketing Management	88	96	92
8	Private Bank	869	1053	800
9	Retail	684	782	672
10	Settlement	156	222	153
11	Foreign Trade	84	111	80
12	Charter Business	48	32	59
13	Innovative Product	21	10	23
14	Communation Platform	18	5	21
15	Joint Card	94	24	92
16	Block Chain	42	10	42
17	Intelligent Answering	8	5	8
18	Data Analysis	33	18	33
19	Anti-Fraud	14	28	13
20	Business Analysis	102	46	75
TOTAL	-	**3,586**	**3,610**	**3,500**

General corpus. For the general corpus, we choose Wikipedia as the data source. Wikipedia is an open knowledge base that covers a large volume of concepts in the real-word, and each concept contains corresponding textual descriptions. To build the general corpus, we randomly sample 12,000 articles from Wikimedia[1] which is a database periodically dumping Wikipedia concepts. We parse the 12,000 articles using an open-source toolkit Wikipedia Extractor[2], and obtain all the textual contents in each concept as the general corpus.

Domain-specific corpus. This domain-specific corpus contains 7,086 requirements across 20 software maintenance projects obtained from company *China Merchants Bank (CMB)*, the largest joint-stock commercial bank in China. CMB's software systems span across a diverse set of banking business, such as debit/credit card, wealth, cash, tax payment, e-billing, etc. Most of CMB's software systems are continuously maintained and evolved over the past 5 years. The 20 projects were completed from January 2018 to June 2019.

Meanwhile, to support requirement measurement and estimation, CMB has maintained a requirement entity list manually. Typically, the development team is responsible for extracting and maintaining the entity list, after passing the review and verification by domain experts. This dual effort is to ensure the quality of the entity list, avoiding reckless errors or individual bias from the engineer teams. However, due to the expensive expert-associated costs, not all requirements will go through this labeling process. More specifically, in this study, this domain-specific corpus consists of two subsets: labeled requirements and unlabeled requirements. The former contains 3,586 requirements produced following the above

[1]https://dumps.wikimedia.org/
[2]https://github.com/attardi/wikiextractor

process. The later contains 3,500 unlabeled requirements. The details of labeled requirements and unlabeled requirements are shown in Table I.

C. Experiment Design

1) Effectiveness Analysis: For RQ1, we firstly pre-train and fine-tune the LSTM language model using all the general corpus and unlabeled requirements respectively, and transfer the trained embedding layer and LSTM layer to the LSTM-CRF model. Finally, we randomly divide the labeled requirements into two parts, i.e., 30% as the training set and remaining 70% as the test set. The training set is used to train the CRF layer in the LSTM-CRF network, and the performance is evaluated on the test set. The last step is repeated 5 times, and the average is used as the final model's performance. Meanwhile, we compare with three baselines to investigate the advantage of RENE. Mann-Whitney tests are conducted between RENE and three baselines respectively to test the differences. Next introduce the baselines.

Baseline 1: AERGT [14]: It is the state-of-the-art approach to automatically extract glossary terms from requirements. AERGT is a heuristics-based approach that does not require the training process and has achieved promising performance in the Satellites and embedded system domains. Following the steps introduced in AERGT, we firstly extract the candidate noun phrases from labeled requirements with the help of an open-source toolkit NLTK[3]. Then, we refine the candidates following the linguistic rules introduced in AERGT.

Baseline 2: Chiu et al.' approach [9]: It is the supervised learning-based approach, which achieves promising performance for the general named entity extraction task. We make use of the implementation published on the Github[4], and train the network using the labeled requirements.

Baseline 3: ACDO [35]: It is the semi-supervised learning-based approach to automatically extract domain-specific terms from natural-language documents, which aims at the insufficient labeled data in a specific domain. Following the approach, we use the labeled requirements as the initial training samples and train an initial CRF model. After that, we implement the iterative process introduced in ACDO to expand training samples from unlabeled requirements and retrain the CRF model until reaching the stop criteria.

2) Transfer Learning Evaluation : For RQ2, we conduct the experiments under the different configurations of RENE:

- $RENE - LM_1 - LM_2$. We remove the pre-training and fine-tuning LSTM language model from RENE and only train all the layers in the LSTM-CRF model with the labeled requirements. Under this setting, the labeled requirements are randomly divided into the training set and test set in the ratio of 3:7. Instead of only training the CRF layer, all three layers in the LSTM-CRF model will be trained with the 30% labeled requirements, and evaluate the performance using the 70% labeled requirements. This setting is used as the comparison benchmark.
- $RENE - LM_2$. Remove Fine-tuning LSTM language model from the RENE. Firstly, we train the LSTM language model using all the general corpus and transfer its embedding layer and LSTM layer to the LSTM-CRF model. Then, we also randomly divide the labeled requirement by the ratio of 3:7. We train the CRF layer using the 30% labeled requirements and evaluate the performance using the remaining 70% labeled requirements.
- $RENE - LM_1$. We train the LSTM language model using all the unlabeled requirements, and transfer layers to the LSTM-CRF model. Then, we only train the CRF layer using 30% randomly divided labeled requirements, and evaluate the performance using the remaining 70% labeled requirements.
- *RENE.* It is same as the experiment design RQ1.

Under each setting, the experiment is repeated 5 times, and the average is used as the final performance.

3) Sensitivity Analysis: For RQ3, we train RENE with different volumes of unlabeled and labeled requirements, and evaluate corresponding performance. First, we use all Wikipedia concepts in the general corpus, and 30% randomly divided labeled requirements to train the LSTM-CRF model. The performance under different sizes of unlabeled requirements is evaluated using the remaining 70% labeled requirements. Second, we take the number of Wikipedia concepts and unlabeled requirements as constant, and randomly sample K% labeled requirements for training and remaining (100-K)% labeled requirements for evaluation. We set K as 10, 20, 30, ..., 90. Both experiments are repeated 5 times respectively, and the average of 5 experiments is used as the final performance.

4) Usefulness Analysis: To answer RQ4, a user study is designed and conducted in September 2019 to evaluate the usefulness of RENE in the real-world application. With the support of our industry collaborator, we apply the RENE in 11 new projects operating on the 20 software systems in CMB, and compare with results produced following the manual labeling process, as introduced in Section IV-B. Specifically, the user study follows four steps: 1) first, we obtain the new requirements of the 11 projects and apply RENE to extract the requirement entities automatically; 2) for each project, our industry collaborator independently collects manually extracted data from corresponding engineering teams who perform entity extraction in the real-world scenario; 3) then, our industry collaborator asks a third-party domain expert to review and verify these two sets of extraction results; and 4) finally, the domain expert's inputs will be used as ground truth, serving for the usefulness evaluation of the proposed RENE, as well as for comparing the results extracted manually and those using RENE.

D. Evaluation Metrics

For all the experiments, we use three commonly-used measurements to evaluate the performance, i.e., *Precision, Recall,*

[3]http://www.nltk.org/

[4]https://github.com/kamalkraj/Named-Entity-Recognition-with-Bidirectional-LSTM-CNNs

F1 [36]. (1) *Precision*, which refers to the ratio of the number of correctly extracted requirement entities to the total number of extracted requirement entities; (2) *Recall*, which refers to the ratio of the number of correctly extracted requirement entities to the total number of requirement entities in the golden test set; and (3) *F1-Score*, which is the harmonic mean of precision and recall.

V. RESULT

A. Baseline Comparison (RQ1)

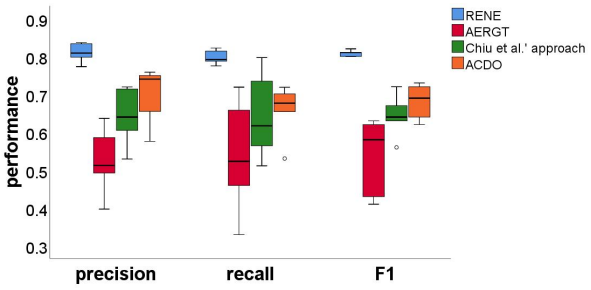

Fig. 3. The box-plot of the performance

Figure 3 shows the box-plot comparison of RENE with three baselines. It is clear that RENE not only achieves the highest performance but also is associated with smaller variation than the baselines. Specifically, RENE reaches 81% precision, 79% recall, and 80% F1 on average, and outperforms all three baselines. The details of average precision, recall, and F1 scores of all four approaches are listed in Table II, along with the results of the Mann-Whitney Test. The numbers in the brackets are the differences from those in RENE. The "*" symbol suffixed with the number indicates that the difference is significant (at the significance level of 0.05).

TABLE II
SUMMARY OF PERFORMANCE COMPARISON RESULTS

approach	precision	recall	F1
RENE	81%	79%	80%
AERGT	46% (-35%*)	60% (-19%*)	51% (-29%*)
Chiu et al.' approach	56% (-25%*)	53% (-26%*)	55% (-25%*)
ACDO	64% (-17%*)	69% (-10%*)	67% (-13%*)

It is surprising to see that the performance of AERGT is the lowest on 20 projects. As a heuristics-based approach, intuitively, AERGT is expected to perform better. When further checking the data, AERGT tends to extract glossary terms that are more general than the requirement entities. For example, when processing a requirement *"As a credit card user, I want to receive the bill reminder message on Android 9.0, so that I could pay back my credit card on time"*, AERGT extracts not only the requirement entity "bill reminder message" but also regular terms such as "credit card", "Android 9.0". This might be due to AERGT's heavy reliance on the linguistic heuristics. The heuristics built on their contexts do not perform well for requirement entity extraction in the 20 projects. Moreover, it is expensive and difficult to rebuild the accurate and complete heuristic rules for the 20 projects which contain abundant business knowledge. As for the recall, RENE also outperforms AERGT, which indicates that RENE could retrieval more requirement entities. Two possible reasons are observed: 1) It is difficult for AERGT to correctly extract long entities, especially for the entities which contain more than 3 words; 2) AERGT does not perform well for the domain-specific entities which are the out-of-vocabulary words for the general NLP techniques. RENE, which is built on the domain-specific contexts and equips with the LSTM-CRF model, could take not only terms themselves but also contextual information into consideration to determine the boundary of a requirement entity, thus could handle the issue better on the 20 projects. Chiu et al.'s approach is a supervised learning approach, which shows poor performance, mainly due to the limited labeled data. RENE also outperforms ACDO that is designed for training the model with a small number of labeled data in the term of semi-supervised machine learning. When expanding the training samples from unlabeled requirements, ACDO introduces false positives that are used are the training samples in the next iteration. The errors accumulate with iterations, which leads to performance degradation. The result indicates that the training procedure in RENE is more effective than the semi-supervised framework used in ACDO.

Summary: RENE significantly outperforms the three state-of-the-art baselines on 20 projects, with performance metrics of 81% precision, 79% recall, and 80% F1.

B. Transfer Learning Evaluation (RQ2)

Table III shows the performance of RENE under different training configurations. The symbol "*" is suffixed with the figure if the Mann-Whitney Test shows the improvement is significant. Overall, compared to $RENE-LM_1-LM_2$ (train the LSTM-CRF model only using labeled requirements), RENE could significantly improve the performance by 11% precision, 12% recall and 12% F1. It is consistent and confirming with the design purpose of introducing the general knowledge to boost the performance of requirement entity extraction.

TABLE III
SUMMARY OF PERFORMANCE UNDER DIFFERENT TRAINING SETTINGS

step	precision	recall	F1
$RENE - LM_1 - LM_2$	69%	72%	70%
$RENE - LM_2$	78% (+9%*)	76% (+4%*)	77% (+7%*)
$RENE - LM_1$	69% (+0%)	74% (+2%)	72% (+2%)
RENE	81% (+12%*)	79% (+7%*)	80% (+10%*)

Compared to the basic configuration ($RENE - LM_1 - LM_2$), each step in RENE improves the performance individually. While the improvement of $RENE - LM_1$ is small and insignificant. The result implies that transfer the layers that are only trained on the unlabeled requirements will not significantly improve the performance. This is likely due to that the unlabeled requirements in our dataset are not enough to capture linguistic knowledge, which emphasizes the importance of large-scale general corpus.

Summary: In general, the training steps in RENE could significantly improve 12% precision, 7% recall, and 10% F1 on the 20 projects. Moreover, each step could significantly boost performance.

C. Sensitivity Analysis (RQ3)

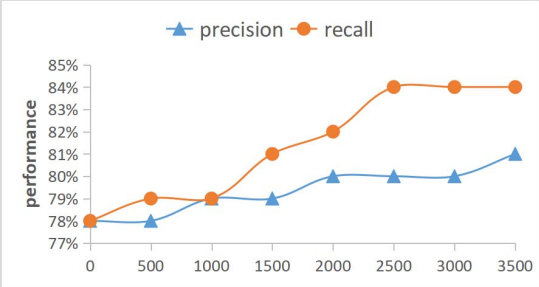

Fig. 4. The performances under different sizes of unlabeled requirements

We investigate the performances under different sizes of unlabeled and labeled requirements. Figure 4 shows the average performance under different sizes of unlabeled requirements. The results of the Mann-Whitney Test between each adjacent sizes indicate that the recall significantly increases before 2,500, and converges after 2500. The precision slowly increases with the number of unlabeled requirements. In general, 2,500 is identified to be the sweet point of cost-effectiveness for unlabeled requirements. Moreover, we can find that the increase in precision looks relatively smaller than recall, which implies that the fine-tuning the LSTM language model primarily contributes to recall.

We use all Wikipedia concepts in the general corpus and 2,500 unlabeled requirements respectively, and change the sampling sizes of labeled requirements. Figure 5 shows the average performances under different sizes of labeled requirements. The results of the Mann-Whitney Test between each adjacent sizes show that both precision and recall significantly increase before 30% (1015) labeled requirements, and there are no significant differences after 30%. Thus, 1,015 is a reasonable number of labeled requirements to train the CRF layer in the LSTM-CRF model.

Summary: In general, 2,500 unlabeled requirements are considered sufficient to adapt the general linguistic knowledge to requirements, and 1,015 is the relatively reasonable number for labeled requirements to train the CRF layer. With 2,500 unlabeled requirements as well as 1,015 labeled requirements, RENE could also achieve promising performance.

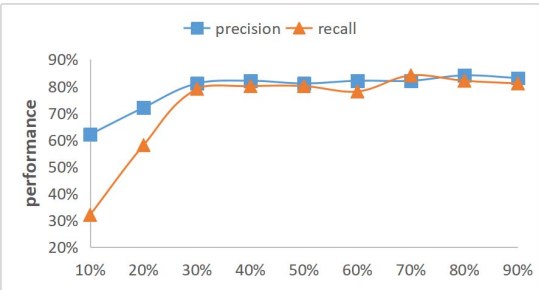

Fig. 5. the performances under different sizes of labeled requirements

D. Usefulness Evaluation (RQ4)

TABLE IV
THE REQUIREMENT ENTITIES EXTRACTED BY ENGINEERS AND RENE

Project	#Req	Expert	Engineer			RENE		
		#Entity	#Entity	P	R	#Entity	P	R
P-1	46	40	45	62%	70%	42	79%	83%
P-2	26	22	26	58%	68%	22	82%	82%
P-3	40	37	41	71%	78%	34	88%	81%
P-4	19	17	18	67%	71%	16	81%	76%
P-5	29	31	29	79%	74%	37	70%	84%
P-6	14	14	15	73%	79%	14	79%	79%
P-7	30	31	32	69%	71%	29	83%	77%
P-8	51	44	48	58%	64%	47	79%	84%
P-9	24	27	24	75%	67%	27	85%	85%
P-10	33	29	33	55%	62%	32	72%	79%
P-11	44	40	44	57%	63%	36	89%	80%
Average	-	-	-	**66%**	**70%**	-	**81%**	**81%**

Table IV shows the results of usefulness evaluation of RENE using the 11 on-going industry projects. The column "Expert" summarizes the ground truth data, i.e. requirement entities manually identified by the domain expert. The column "Engineers" illustrates the manually extracted data from corresponding engineering teams. The column "RENE" shows the requirement entities automatically extracted by RENE, along with its performance metrics of "precision" and "recall". The results show that in the real industrial setting, the quality of manual extraction is not promising (66% precision and 70% recall). In comparison, the results of the Mann-Whitney Test show that the differences between "Engineer" and RENE are significant for both "precision" and "recall". The results indicate that RENE could produce more accurate and completed requirement entities for almost all the projects. By analyzing the results of requirement entities extracted by engineers, we find that the major reason for the differences in both extractions is that RENE tends to extract more complete phrases that describe the particular business concepts, while engineers tend to extract partial phrases, esp. for rather long, business-specific entities. For example, there is a sentence *"As a business manager, I would like to automatically load the merchant loan information template when the merchant fills in the loan information so that the merchant can fill in the relevant information in the specific format."* The requirement entity extracted by the engineer is *"loan information template"*, while the ground truth is *"the merchant loan information template"*. It corresponds to a loan form specific for the merchants rather than personals in CMB. This indicates that manual labeling is prone to extracting incomplete expressions, leading to ambiguity and affect subsequent software development. By comparison, RENE could handle the problem better and extract the complete requirement entities in practices.

Summary: When applied to 11 on-going software projects, RENE achieves the precision and recall both at 81% for

automated requirement entity extraction, and significantly outperform the manual extraction by the engineers. The results indicate that the more accurate and complete requirement entity list for each project could be built with the help of RENE.

VI. DISCUSSION AND THREATS

A. Benefits

In the application scenario, there are normally two situations for a continuous-evolved software system: 1) a requirement leads to new functionality that needs to be added into the system; 2) a requirement involves the changes of the existing functionality in the system. For the first case, the requirement entities extracted by RENE could be used as the sources to build the feature model, design the architecture of code and database schema, expand the system knowledge base and etc [10], [37], [38]. For the second case, there are typically existing entities in the systems. RENE could also be applied to extract the requirement entities. The extracted requirement entities could be linked into different elements as entities in a feature model, code entities, database entities, and knowledge bases using the Entity Linking techniques [39]–[41] for further change impact analysis [6] and feature location [42].

B. Lessons Learned

1) Handling the out-of-vocabulary problem: In our practice, we found that a large number of textual requirements are incorrectly parsed by NLP tool kits. After analyzing the incorrect cases, we noted that most cases are related to business terms which are named by the company according to the specific business scenarios. It is known as the out-of-vocabulary (OOV) problem [17] in NLP. This is due to that the tool kits are all trained on the general corpus which mainly contain a common vocabulary of terms. Especially, the OOV problem is particularly obvious in the languages which are not naturally segmented by spaces, such as Chinese, Korean, and Japanese. The errors of NLP tool kits can be introduced into the approaches built upon the general NLP techniques, which could also influence their performance. To solve the OOV problem, almost all NLP tool kits have provided interfaces to import custom dictionary to help parse the texts. However, this solution requires lots of human efforts to carefully review a large number of documents to ensure the high quality of dictionaries, which is cost-consuming. Another solution is to leverage the learning-based approaches, which are not totally built upon NLP tool kits, and has been verified to handle the OOV problem in NLP tasks effectively [43]–[45].

2) Sequence modelling instead of bag-of-words modeling: When dealing with tasks such as text classification, information retrieval, these natural language artifacts are usually modeled using bag-of-words (BOW) [46]. BOW puts all words in a bag, regardless of their morphology and word order, that is, each word is independent. Considering that natural language is a sequence of words, in which each word carries rich grammatical and contextual information, it is a better choice to model the natural language articles into sequences rather than BOW. Especially for the requirement entities, they typically appear with indicative contexts, which makes the sequence tagging model such as the LSTM-CRF model achieve satisfactory results.

3) Dealing with tasks with limited labeled data: Thanks to the advances of machine learning techniques, many related tasks have now reached impressive performance [47]–[49]. However, they are mostly supervised learning which requires massive labeled data, incurring significant labeling effort [12]. Compared to labeled data, it is much easier to obtain an amount of historical data without labeling. In addition, there are also lots of freely accessible data sources such as Wikipedia, StackOverflow, and Github, which have been applied to improve the performances of specific tasks such as software text retrieval [50], requirement term extraction [51], and text classification [52]. In our practice, we propose a novel transfer learning approach that utilizes Wikipedia concepts and unlabeled requirements to boost performance. We consider the proposed approach is not limited to requirement entity extraction, and could be generalized to more application scenarios with low labeled resources. Therefore, when we only have limited labeled resources, it is a potential choice to consider using public corpora and unlabeled data to improve the performance.

C. Threats to Validity

External Validity. The external threats are related to the generalization of the proposed approach. First, we experimented with the data taken from a company. The results may be different from other scenarios. However, CMB is a large financial company, and there are over 200 teams that cover different business applications. In addition, in our case study, we train the model and evaluate the performance on the requirements from 20 systems, which could reduce this threat. Second, the studied subjects in this study are requirements only, which may not be appreciated to other artifacts like app reviews and code documentation. However, RENE extracts requirement entities from sentences and do not utilize unique characteristics except for natural language description, which could alleviate the threat.

Internal Validity. The internal threats relate to experimental errors and biases. First, in our approach, we utilize the unlabeled requirements aiming at adapting to characteristics of requirements. These unlabeled requirements are given by CMB in the financial domain, and the adapted knowledge model may also be restricted to the financial domain rather than general requirements. However, there are 3,500 requirements from 20 projects used for fine-tuning the LSTM language model. Each project contains its topics and applications, which could alleviate the threats. Second, the AERGT is designed for requirements glossary term extraction, and is included as the baseline approach for requirement entity extraction. Considering that AERGT is conducted under different contexts, it may introduce bias into the performance.

VII. RELATED WORK

We introduce the related approaches for term extraction. According to the techniques used in the approaches, we divide them into two categories, i.e., heuristics-based approaches and learning-based approaches.

A. Heuristics-Based Approaches

Typically, heuristics-based approaches parse the text candidates using NLP tool kits, then build linguistic rules upon the parsing results according to expertise to extract the target information. To our best knowledge, all the previous studies for requirements belong to this group. Bourigault et al. [53] described a linguistic approach for terminological noun phrases. It is implemented by regular expressions along with part-of-speech tags returned by the NLP toolkit to extract certain combinations of noun phrases. Dwarakanath et al. [54] used linguistic rules to identify process nouns, abstract nouns, and auxiliary verbs from natural language requirements. It firstly parses part-of-speech tags, and builds the linguistic rules to handle co-ordinating conjunctions, adjectival modifiers and nominalizations. Then it refines the results by statistical techniques. Ménard et al. [55] proposed an approach to extract domain-specific concepts from business documents. It is implemented as a pipeline including a candidate generation step based on part-of-speech patterns, several filtering modules to filter out the irrelevant terms by heuristics. Johann et al. [5] proposed an approach to extract noun phrases verb phrases that are related to features from app descriptions and app reviews. It gives 18 predefined part-of-speech patterns to extract phrases. Arora et al. [14] developed an approach for extracting glossary terms and their related terms from requirements documents. It firstly extracts the candidate noun phrases via NLTK. Then, three linguistic heuristics are utilized to refine the candidates. These approaches typically give a set of pre-defined rules according to the expertise in their contexts. Different from the previous studies, our approach is specific to the requirement entity extraction and based on the machine learning model which could avoid the overload of manually building expert rules for each domain.

B. Learning-Based Approaches

Learning-based approaches adopt the machine learning algorithm to extract target information from texts, have been widely-used for many general information extraction tasks. Finkel et al. [48] trained a CRF model for named entity recognition (NER) with 10 hand-crafted features. Lample et al. [8] presented a widely-used LSTM-CRF model for NER, which aims at extracting target information in terms of sequence tagging and eliminates the need for feature engineering. Chiu et al. [9] presented a novel neural network architecture to extract named entities from the general corpus. It detects word-level and character-level features using a hybrid bidirectional LSTM and CNN architecture. With the help of machine learning, general information extraction tasks have achieved promising performance under the condition of sufficient labeled data. However, labeling is a labor-intensive activity especially in the domains requiring extensive expertise. Thus, a growing number of researches focused on boosting performance by unlabeled data. Huang et al. [35] proposed an approach to extracted domain-specific concepts based on a semi-supervised CRF model. Using a small number of labeled data as seeds, it iteratively expands training samples from unlabeled data by bootstrapping. It is included in our study as the baseline approach. Sachan et al. [56] investigated how to use unlabeled text data to improve the performance of named entity recognition. Specifically, they trained a bidirectional language model (BiLM) on unlabeled data and transfer its weights to pre-train a named entity recognition model with the same architecture as the BiLM, which results in a better parameter initialization of the NER model. Chong et al. [10] proposed a semi-supervised approach based on LSTM-CRF model for the glossary construction from source code and documentation. It firstly identifies the initial identifiers from the source code via heuristics. Using the initial identifiers as the seed terms, it iteratively expands the training set from unlabeled comments and trains the LSTM-CRF model via bootstrapping. Different from the previous studies which focus on general corpus, our study is aimed at requirement entity extraction in specific domains. The goal of the proposed approach is to solve the problem of lacking domain-specific NLP techniques

VIII. CONCLUSION

Establishing and maintaining requirement entities plays an important role in software development and maintenance. This paper proposed a novel approach RENE to extract requirement entities from textual requirements automatically. It employs the LSTM-CRF model and transfer learning to resolve the domain-specific problem and data deficiency issues. We evaluated the RENE on an industrial dataset build from 20 continuously-evolved software systems. The evaluation results show that the proposed approach could reach 81% precision, 79% recall and 80% F1, and outperforms the three state-of-the-art baselines. And we verified the value of general corpus and unlabeled data and provided an effective practice to make use of these data, which could be an inspiration on how to further boost the performance for the specific task. Moreover, by expert review on 11 randomly sampled projects, our approach could produce more accurate and complete requirement entities than the manual extraction by engineers. The presented material is just the starting point of the work in progress. Future work will also include similar requirement entity clustering and automatic requirement entity-relationship construction.

ACKNOWLEDGMENT

This work is supported by China Merchants Bank, the National Science Foundation of China under grant No.61802374, No.61432001, No.61602450, and the National Key Research and Development Program of China under grant No.2018YFB1403400.

REFERENCES

[1] J. E. Matson, B. E. Barrett, and J. M. Mellichamp, "Software Development Cost Estimation Using Function Points," *IEEE Transactions on Software Engineering*, vol. 20, no. 4, pp. 275–287, 1994.

[2] M. Bundschuh and C. Dekkers, *The IFPUG Function Point Counting Method*. Springer Berlin Heidelberg, 2008.

[3] A. Hira and B. W. Boehm, "Function point analysis for software maintenance," in *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2016, Ciudad Real, Spain, September 8-9, 2016*, 2016, pp. 48:1–48:6.

[4] A. Z. Abualkishik, F. Ferrucci, C. Gravino, L. Lavazza, L. Geng, R. Meli, and G. Robiolo, "A study on the statistical convertibility of ifpug function point, cosmic function point and simple function point," *Information & Software Technology*, vol. 86, pp. 1–19, 2017.

[5] T. Johann, C. Stanik, W. Maalej *et al.*, "Safe: A simple approach for feature extraction from app descriptions and app reviews," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE, 2017, pp. 21–30.

[6] W. Khlif, M. Haoues, A. Sellami, and H. Ben-Abdallah, "Analyzing functional changes in bpmn models using cosmic," in *12th International Conference on Software Technologies*, 2017.

[7] W. Khlif, A. Sellami, M. Haoues, and H. Ben-Abdallah, "Using cosmic fsm method to analyze the impact of functional changes in business process models," in *Enase: International Conference on Evaluation of Novel Approaches to Software Engineering*, 2018.

[8] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, 2016, pp. 260–270.

[9] J. P. C. Chiu and E. Nichols, "Named entity recognition with bidirectional lstm-cnns," *Computer Science*, 2016.

[10] W. Chong, P. Xin, L. Mingwei, X. zhenchang, B. Xuefang, X. Bing, and W. Tuo, "A learning-based approach for automatic construction of domain glossary from source code and documentation," in *ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering(ESEC/FSE)*, 2019.

[11] X. J. Zhu, "Semi-supervised learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2005.

[12] V. T. Dhinakaran, R. Pulle, N. Ajmeri, and P. K. Murukannaiah, "App review analysis via active learning," in *International Requirements Engineering Conference*, 2018.

[13] C. Arora, M. Sabetzadeh, L. C. Briand, and F. Zimmer, "Improving requirements glossary construction via clustering: approach and industrial case studies," in *Acm/ieee International Symposium on Empirical Software Engineering and Measurement*, 2014.

[14] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer, "Automated extraction and clustering of requirements glossary terms," *IEEE Transactions on Software Engineering*, vol. 43, no. 10, pp. 918–945, 2016.

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

[16] T. Gemkow, M. Conzelmann, K. Hartig, and A. Vogelsang, "[ieee 2018 ieee 26th international requirements engineering conference (re) - banff, ab, canada (2018.8.20-2018.8.24)] 2018 ieee 26th international requirements engineering conference (re) - automatic glossary term extraction from large-scale requirements," pp. 412–417, 2018.

[17] L. Qin, "Learning out-of-vocabulary words in automatic speech recognition," Ph.D. dissertation, Citeseer, 2013.

[18] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification," *Lingvisticæ Investigationes*, vol. 30, no. 1, pp. 3–26, 2007.

[19] E. F. T. K. Sang, "Introduction to the conll-2002 shared task: Language-independent named entity recognition," 2002.

[20] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," 2001.

[21] Sutton and Charles, "An introduction to conditional random fields," *Foundations & Trends® in Machine Learning*, vol. 4, no. 4, pp. 267–373, 2012.

[22] A. Tang, D. Jackson, J. Hobbs, W. Chen, J. L. Smith, H. Patel, A. Prieto, D. Petrusca, M. I. Grivich, and A. Sher, "A maximum entropy model applied to spatial and temporal correlations from cortical networks in vitro," *Journal of Neuroscience the Official Journal of the Society for Neuroscience*, vol. 28, no. 2, pp. 505–518, 2008.

[23] J. Felsenstein and G. A. Churchill, "A hidden markov model approach to variation among sites in rate of evolution," *Molecular Biology & Evolution*, vol. 13, no. 1, pp. 93–104, 1996.

[24] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF models for sequence tagging," *CoRR*, vol. abs/1508.01991, 2015.

[25] Z. Jie and W. Lu, "Dependency-guided LSTM-CRF for named entity recognition," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, 2019, pp. 3860–3870.

[26] M. Xu, X. Zhang, and L. Guo, "Jointly detecting and extracting social events from twitter using gated bilstm-crf," *IEEE Access*, vol. 7, pp. 148 462–148 471, 2019.

[27] S. Ceri, A. Bozzon, M. Brambilla, E. D. Valle, P. Fraternali, and S. Quarteroni, "An introduction to information retrieval," *Pharmacogenomics Journal*, vol. 2, no. 2, pp. 96–102, 2013.

[28] R. Rosenfeld, "Two decades of statistical language modeling: Where do we go from here?" *Proceedings of the IEEE*, vol. 88, no. 8, pp. 1270–1278, 2000.

[29] S. J. Pan and Y. Qiang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[30] R. Socher, Y. Bengio, and C. D. Manning, "Deep learning for nlp (without magic)," *Acl Tutorial*, 2013.

[31] R. Zunino and P. Gastaldo, "Analog implementation of the softmax function," in *IEEE International Symposium on Circuits & Systems*, 2002.

[32] N. R. Pal, "On minimum cross-entropy thresholding," *Pattern Recognition*, vol. 29, no. 4, pp. 575–580, 1996.

[33] L. Ratinov and R. Dan, "Design challenges and misconceptions in named entity recognition," in *Conll 09: Thirteenth Conference on Computational Natural Language Learning*, 2009.

[34] H. J. Dai, P. T. Lai, Y. C. Chang, and T. H. Tsai, "Enhancing of chemical compound and drug name recognition using representative tag scheme and fine-grained tokenization," *Journal of Cheminformatics*, vol. 7, no. S1, p. S14, 2015.

[35] R. Huang and E. Riloff, "Inducing domain-specific semantic class taggers from (almost) nothing," 2010.

[36] D. M. Berry, J. Cleland-Huang, A. Ferrari, W. Maalej, J. Mylopoulos, D. Zowghi, D. M. Berry, J. Cleland-Huang, A. Ferrari, and W. Maalej, "Panel: Context-dependent evaluation of tools for nl re tasks: Recall vs. precision, and beyond," in *Requirements Engineering Conference*, 2017.

[37] D. Ye, Z. Xing, C. Y. Foo, Q. A. Zi, L. Jing, and N. Kapre, "Software-specific named entity recognition in software engineering social content," in *IEEE International Conference on Software Analysis*, 2016.

[38] X. Zhao, Z. Xing, M. A. Kabir, N. Sawada, and S. W. Lin, "Hdskg: Harvesting domain specific knowledge graph from content of webpages," in *IEEE International Conference on Software Analysis*, 2017.

[39] C. L. Tan, "Entity linking leveraging automatically generated annotation," in *The 23(rd) International Conference on Computational Linguistics Proceedings of the Main Conference (Volume 2)*, 2010.

[40] X. Han, S. Le, and J. Zhao, "Collective entity linking in web text," in *Proceeding of International Acm Sigir Conference on Research & Development in Information Retrieval*, 2011.

[41] W. Shen, J. Wang, and J. Han, "Entity linking with a knowledge base: Issues, techniques, and solutions," *IEEE Transactions on Knowledge & Data Engineering*, vol. 27, no. 2, pp. 443–460, 2015.

[42] F. Pérez, J. Font, L. Arcega, and C. Cetina, "Collaborative feature location in models through automatic query expansion," *Autom. Softw. Eng.*, vol. 26, no. 1, pp. 161–202, 2019.

[43] W. Ling, I. Trancoso, C. Dyer, and A. W. Black, "Character-based neural machine translation," *arXiv preprint arXiv:1511.04586*, 2015.

[44] A. Maas, Z. Xie, D. Jurafsky, and A. Ng, "Lexicon-free conversational speech recognition with neural networks," in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015, pp. 345–354.

[45] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4960–4964.

[46] A. Sethy and B. Ramabhadran, "Bag-of-word normalized n-gram models," in *Ninth Annual Conference of the International Speech Communication Association*, 2008.

[47] A. Di Sorbo, S. Panichella, C. V. Alexandru, J. Shimagaki, C. A. Visaggio, G. Canfora, and H. C. Gall, "What would users change in my app? summarizing app reviews for recommending software changes," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 2016, pp. 499–510.

[48] J. R. Finkel, T. Grenager, and C. Manning, "Incorporating non-local information into information extraction systems by gibbs sampling," in *Meeting on Association for Computational Linguistics*, 2005.

[49] W. Maalej, Z. Kurtanović, H. Nabil, and C. Stanik, "On the automatic classification of app reviews," *Requirements Engineering*, vol. 21, no. 3, pp. 311–331, 2016.

[50] Z. Lin, Y. Zou, J. Zhao, and B. Xie, "Improving software text retrieval using conceptual knowledge in source code," in *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2017, pp. 123–134.

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

[52] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," 2018.

[53] D. Bourigault, "Surface grammatical analysis for the extraction of terminological noun phrases," in *Proceedings of the 14th conference on Computational linguistics-Volume 3*. Association for Computational Linguistics, 1992, pp. 977–981.

[54] A. Dwarakanath, R. R. Ramnani, and S. Sengupta, "Automatic extraction of glossary terms from natural language requirements," in *2013 21st IEEE International Requirements Engineering Conference (RE)*. IEEE, 2013, pp. 314–319.

[55] P. A. Ménard and S. Ratté, "Concept extraction from business documents for software engineering projects," *Automated Software Engineering*, vol. 23, no. 4, pp. 649–686, 2016.

[56] D. S. Sachan, P. Xie, and E. P. Xing, "Effective use of bidirectional language modeling for medical named entity recognition," *CoRR*, vol. abs/1711.07908, 2017.