CrowDevBot: A Task-Oriented Conversational Bot for Software Crowdsourcing Platform

Zeyu Ni, Zhangyuan Meng, Junming Cao, Beijun Shen*, Yuting Chen
School of Electronic Information and Electrical Engineering
Shanghai Jiao Tong University, Shanghai, China
{sfordj.ni, m602389789, junmingcao, bjshen, chenyt}@sjtu.edu.cn

***Abstract*—With the trends of developing software on the Internet, many software crowdsourcing platforms are emerging. They attract a lot of developers to bid for crowdsourced projects and develop software systems collaboratively. In this paper, we present CrowDevBot, *a task-oriented conversational bot for software crowdsourcing platform*, that aims to assist online users in completing crowdsourcing-related tasks in a more natural manner. The key idea of CrowDevBot is to: (1) combine a rule-based method and an SVM-NaiveBayes-C4.5 integrated learning method to discover users' intention; (2) employ an integrated CRF (conditional random field) method with novel features to improve the performance of slot filling; and (3) leverage a software service knowledge base to unify entity names and predefine the key slots of user query. We implement CrowDevBot and integrate it into JointForce, an IT software crowdsourcing platform in China. To the best of our knowledge, this is the first time that a task-oriented conversational bot is practically used in software crowdsourcing platform(s). We evaluated our approach on real data set from JointForce. The results show that our intention detecting method achieves F1-score of 87% on the limited training data. For the slot filling, the F1-score of our integrated CRF model reaches 82%, 8% higher than that of the normal CRF model.**

***Index Terms*—Task-oriented conversational bot, software crowdsourcing platform, integrated statistical learning, user intention understanding**

I. Introduction

Software crowdsourcing [1] is a new software development paradigm. In a crowdsourcing process, project requesters and software developers need to complete many tasks, such as recommending qualified developers, searching and biding projects, querying project progresses, evaluating service qualities, etc. This paper presents *a task-oriented conversational bot* (CrowDevBot) for software crowdsourcing platform. A bot can assist users in completing their crowdsourcing-related tasks through conversation, rather than tedious mouse clicks. Thus it supplements crowdsourcing platforms with strong flexibility. Figure 1 demonstrates how such a bot helps an engineer to complete his/her task.

Though developinging bots can benefit crowdsourcing platforms in completing software crowdsourcing tasks in a more friendly manner, it still faces three challenges:

Challenge 1. *How is the cold start problem solved by a task-oriented conversational bot?* Different from online-shopping websites, user interactions on software crowdsourcing platform is much less, which brings the cold-start problem. Statistical learning models or deep learning models may also not work well due to limited data.

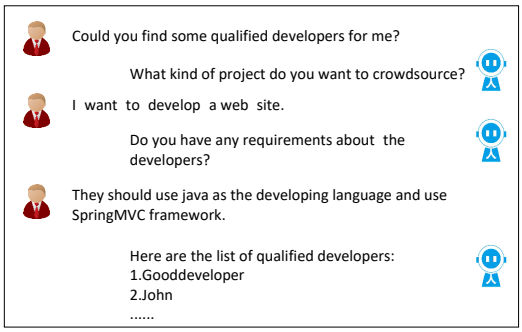

Fig. 1: An example of using CrowDevBot in crowdsourcing

To solve the cold start problem, several rule-based methods have been proposed for natural language understanding [2] [3] [4]. However, rule-based methods usually have following shortcomings: (1) The accuracy of understanding users' intention relies on the quality of rules. (2) User queries are usually flexible while a pre-defined rule set may not be complete.

Challenge 2. *How is the performance of CRF model improved for user intention understanding?*

CRF (Conditional Random Field) is a widely used model for solving the slot filling problem of user intention understanding. Researches [5] [6] show that CRF can be a general model, while many new deep learning models are not [7] [8] [9].

Due to limited training data, the normal CRF is still unsatisfactory. To improve the slot filling performance of CRF model, there are two key issues: (1) how to define effective features according to the characteristics of software crowdsourcing domain; and (2) how to design an improvement strategy to achieve a more robust CRF model.

Challenge 3. *How can the software service knowledge base be utilized to enhance the capability of the bot?*

Priori knowledge on software crowdsourcing can definitely enhance the capability of CrowDevBot. Inspired by the work of Zhao et al. [10], we build a knowledge base to represent software service knowledge. As far as we know, this is the first time that a task-oriented conversational bot is practically used

*Corresponding author
DOI reference number: 10.18293/SEKE2019-068

in software crowdsourcing scenario. So it will be challenging for us to design and leverage the software service knowledge base in CrowDevBot, to improve its performance.

To address these challenges, we propose a novel approach to developing CrowDevBot. CrowDevBot consists of five key components: *user intention detecting*, *slot filling*, *dialog management*, *task execution*, and *answer generation*. We combine the rule-based method and a SVM-NaiveBayes-C4.5 integrated learning method to deal with the cold start problem when detecting user intentions. CrowDevBot adaptively sets weights for these two methods. For filling slots, we propose an integrated CRF model with novel features, including syntactic and semantic features. We also leverage a software service knowledge base to predefine the key slots of software services and normalize their entity names. So far, we have integrated CrowDevBot into JointForce, one of the biggest IT software crowdsourcing platforms in China. To the best of our knowledge, this is the first time that a task-oriented conversational bot is used in software crowdsourcing platform.

II. Related Work

Storey and Zagalsky [11] propose that bots can act as "conduits between users and services, typically through a conversational UI". Roughly bots can be divided into two categories [12] : chat-oriented bots and task-oriented bots. In recent years some task-oriented conversational bots have been built both in industry (such as Apple's Siri, Microsoft's Cortana) and in academia (such as those proposed by Zhou et al. [10], and Wen et al. [13]). Bots are also rapidly becoming a general interface for software services communication [14]. However, to our best knowledge, few task-oriented conversational bots have been developed for software crowdsourcing platforms.

User Intention Detecting. User Intention Detecting is an important part of task-oriented bots, which can be seen as a classification problem. Particularly, more and more deep models are proposed for this purpose. Kim Y et al. [15] first apply the CNN (convolution neural network) model to solving these problems. Xu et al. [16] use RNN (recurrent neural network)) to detect user intentions. Compared to the CNN model, RNN makes use of the sequence information, making it fit for natural language problem. However, no matter how novel the network structure, the performance is poor when data is limited. It requires bots to handle this problem.

Slot Filling. After user intention is detected, the bot needs to parse the user input and recognize the predefined key slots (words to be filled in a sentence). It can be taken as a named entity recognition problem. Mccallum [5] et al. use the traditional CRF to solve this problem. Though this model works, it is unable to handle the OOV problem, i.e., it cannot recognize an entity which do not appear in the training data. Thus more novel features need to be designed. Researches [17] apply deep learning methods on solving this problem. But as mentioned before, the restrictions on data reduce the performance of these models.

Cold-Start Problem. When a task-oriented conversational bot is built, it faces with the potential cold-start problems. The cold-start problem is serious for the software crowdsourcing platform as the user data is limited [18]. To solve the cold-start problem in a universal way, Rieser V et al. [4] focuse on developing a structured ontology for parsing utterance from user into predefined semantic slots. Zhao Yan et al. [10] present a general solution to the cold-start problem in online-shopping domain. They use crowdsourcing to label training data, while it is not suitable for our situation as data is limited.

III. Approach

A. Approach Overview

Our approach overview is shown in Figure 2. The task-oriented conversational bot for software crowdsourcing platform, named CrowDevBot, consists of 5 components.

1) User Intention Detecting: Given a user query, CrowDevBot detects the user intentions with a combination method after entity name unification. A rule-based method and an SVM-NaiveBayes-C4.5 integrated learning method is combined by the following strategy: the rule based method is mainly used in the CrowDevBot's startup process; with data increases, the SVM-NaiveBayes-C4.5 integrated learning method becomes more effective, and thus its weight gets increased.

2) Slot Filling: This component gathers the key information (called "slot") from user queries. We design several key features (including transition feature, start and end features, word and syntactic features), and employ integrated CRF model to fill slots together with probability distribution information to achieve better precision and robustness.

3) Dialogue Management: In this component, the interaction process between user and CrowDevBot is managed. CrowDevBot uses a FSM (Finite State Machine) to track the dialogue states. When some slots are missing, it launches new questions to ask. We utilize the software service knowledge base (SSKB) to predefine the key slots for software services. When all of the slots are filled, CrowDevBot invokes *task executing component* to execute the task.

4) Task Executing: In this component, the user tasks are executed: once CrowDevBot has identified user intention and filled all slots, it will invoke the corresponding APIs of software crowdsourcing platform and return results to CrowDevBot.

5) Answer Generation: As all the supporting tasks are clear and well defined, it can be easy to use a template library to generate answers. CrowDevBot uses FSM to determine which template to be used to generate answers or launch new questions.

Next explains the details of the software service knowledge base, user intention detecting and slot filling.

B. Software Service Knowledge Base

To leverage the domain knowledge of software crowdsourcing, we build a software service knowledge base (SSKB). The main data sources of SSKB includes the knowledge

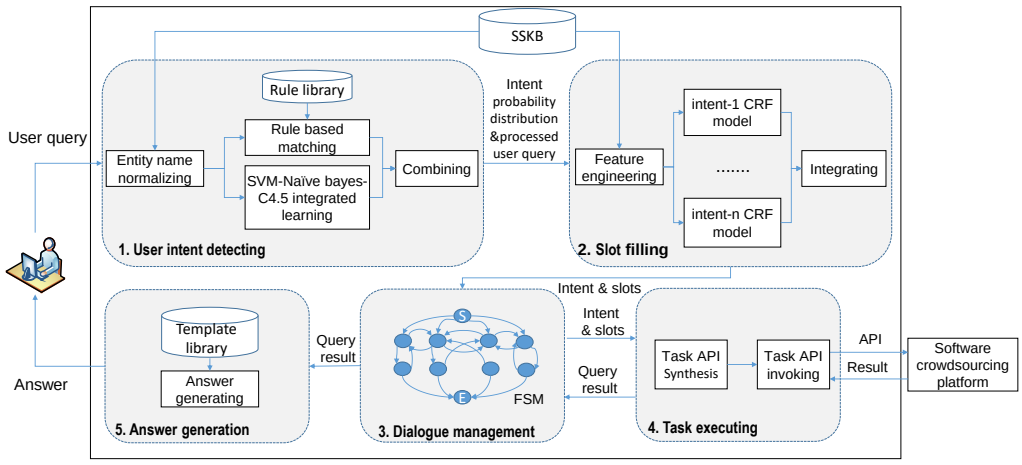

Fig. 2: A general process of CrowDevBot in processing user queries in a software crowdsourcing platform

collected from software crowdsourcing experts, StackOverflow tag synonym system* and Wikipedia†. We apply NLP technology to extract information from these (semi-)structured data, and construct SSKB, which contains category system, technologies, and attributes of software services. For more details, see our project in github‡.

We exploit SSKE to improve the performance of CrowDevBot in two scenarios. The first scenario is entity name unification. In the dialogue with CrowDevBot, users tend to express casually, with various aliases and abbreviations, leading to decreases in precision of intention detecting and slot filling. Thus we build a synonym dictionary using entity synonym attributes in SSKE. During preprocessing, CrowDevBot replaces aliases and abbreviations with standard names.

The second scenario exists in defining slots for software services. For example, in Figure 1, after CrowDevBot receives user input "I want to develop a web site", it maps the query to the entity "web development" in SSKE. Thus CrowDevBot continuously asks user about the attributes values of "web development" to complement the corresponding slots.

C. User Intention Detecting

User intention detecting can be formulated as a user query classification problem. To solve it, we propose a mixture method, which combines a rule based method and an SVM-NaveBayes-C4.5 integrated learning method. Weights are dynamically assigned to these two methods. In the startup process, CrowDevBot relies mainly on the rule based method, as it requires less usable training data. After sufficient data is collected, the SVM-NaveBayes-C4.5 integrated learning method will make more contribution.

1) Rule-Based Method: Each kind of intention corresponds to a set of query rules that are used to describe the possible ontology structures of user queries. A rule consists of several tokens. Each token has 3 attributes: Match_pattern, Weight and Indispensable. Here Match_pattern contains its candidate words, Weight shows the importance of the token to the whole rule, and Indispensable explains the necessities of a token.

The matching algorithm is designed as Algorithm 1 shows.

Algorithm 1 Greedy rule matching algorithm.

Require: A rule (R) and word list (QL) of user query
Ensure: Matching degree (MD) between the rule and query

1: **for** each $word\ or\ parse \in QL$ **do**
2: **for** each $unmatched\_tokenT \in R$ **do**
3: compute the word2vec vector similarity between w and each word in $T.Match\_pattern$, and record the highest HS_w
4: **if** $HS > 0.5$ **then**
5: mark this token as *matched_token* and record the similarity
6: goto step 1
7: **if** all tokens in $(R.token|R.token.indispensable = 1)$ are matched **then**
8: $MD = \frac{\sum_{a \in matched\_token} token_a.weight \times HS_a}{\sum_{b \in R.token} token_b.weight}$
9: **else**
10: $MD = 0$
11: **return** MD

2) Integrated Learning Method: CrowDevBot faces with the cold start problem, making deep learning techniques inappropriate to use. Thus we propose an integrated statistical learning method to train the intention classification model. We select SVM, Naive Bayes and C4.5 DT as basic models and combine them following two strategies: Bagging and AdaBoost.

We design three features after manual analysis on these wrong detection cases.

- Semantic feature. The feature is produced by POS tagging. The possible values includes noun, adjective, etc.

*http://stackoverflow.com/tags/synonyms
†https://www.wikipedia.org
‡https://github.com/SE1405Lab/SSKB

- N-gram feature. The n-gram feature [19] takes the word sequence-level information to help analyze a sentence. In our method, we take N as 3.
- Word2vec feature. The Word2Vec feature [20] represents words with a low-dimentional vector, which helps understand the semantics of words.

3) Method Combination: The results of these two methods above are combined with weights:

$$R = W_1 R_1 + W_2 R_2, \tag{1}$$

where $W1$ is the weight of rule-based method, and $R1$ is the match degree between user query and each intention; $W2$ is the weight of SVM-NaiveBayes-C4.5 method, and $R2$ is its probability distribution. The intention with the highest R will be regarded as the final output.

D. Slot Filling

We use CRF [6] as a basic model to fill the slots, as CRF achieves better performance than general models (like HMM [21]). Given the word sequence $X = (x_1, x_2, ..., x_m)$, CRF computes the conditional probability of a label sequence. It produces a label $y = (y_1, y_2, ..., y_m)$ sequence to maximize $p(y|x)$.

$$p(y|x) = \frac{1}{z_\lambda(x)} exp\{\lambda \cdot f(y, x)\} \tag{2}$$

Meanwhile, it is intractable to compute $f(y, x)$. Thus we assume that the value of $f(y, x)$ depends only on the adjacent labels and the formula $p(y|x)$ can be converted into

$$p(y|x) = \frac{1}{z_\lambda(x)} exp\{\sum_{i=1}^{M+1} \lambda \cdot f(y_{i-1}, y_i, x, i)\} \ , \tag{3}$$

where $z_\lambda(x)$ is a normalization factor, and λ the weight vector of feature functions. The training process aims to find a proper λ for the CRF model.

We design several types of features for CRF:

- Transition feature: A transition feature describes the transition between adjacent labels.
- Start feature: A start feature implies that a label happens to be at the start position.
- End feature: An end feature implies that a label happens to be at the end position.
- Word feature: A word feature represents the co-occurrence of a word and a label.
- Syntactic feature: A syntactic feature represents the co-occurrence of a POS tag of the word and a label.
- Semantic feature: Inspired by Li et al.'s work [22], we design the semantic feature using "lexicons", which are clusters of semantically-related word or phrase constructs. A semantic feature describes the co-occurrence of an element in a lexicon L and a label.

We train several CRF models for each intention and integrate them into one mixture model to fill slots of templates. After the slot values are obtained, CrowDevBot uses SSKE to examine whether the key slots are complete and then traces the states using the FSM.

TABLE I: Intention detecting results w.r.t. different models

Model	Precision	Recall	F1-score
Integrated (Bagging)	0.78	0.85	0.80
Integrated (Boosting)	0.83	0.89	0.83
SVM	0.77	0.81	0.78
Naive Bayes	0.65	0.73	0.68
DT(C4.5)	0.75	0.75	0.75
RNN	0.76	0.82	0.78

TABLE II: Intention detecting results with different methods

Method	Precision	Recall	F1-score
Rule Based	0.90	0.61	0.73
Integrated Learning	0.83	0.89	0.83
Combination	0.90	0.85	0.87

IV. Experiments

We evaluate our bot on real data from JointForce.

A. Experiment Setup

Three evaluation metrics are selected: Precision, Recall and F-1 score. We have collected 1458 user queries from JointForce[§]. These queries are classified into 6 categories, in each of which we picked up 200 queries with slot information and labeled them manually. We also asked Chinese linguists to design 65 rules such that our rule-based method can be applied.

B. Intention Detecting Experiments

We compared the effectiveness of each single statistical learning model and the integrated model in detecting user intentions. Due to the small amount of our data, we used the BootStrapping method to obtain the training and testing dataset.

The results are shown in Table I. The integrated statistical learning model (with the Boosting training strategy) achieves the highest precision and recall. On the contrary, the deep models are not well supported by a small scale dataset, leading to lower precision and recall.

Next we compared the effectiveness of the rule based method, the integrated statistical learning method and the combination method. The results, as Table II shows, denote that the combination method is slightly better than the statistical learning method. The rule-based method achieves a high precision, indicating the benefits from strictness. The rule matching algorithm implies that, once the user query is matched with one rule, it is much possible to be a true positive.

C. Slot Filling Experiments

We analyzed the feature contributions to slot filling, through an experiment that applys our integrated CRF model with different feature sets. The result is shown in Table III, where iCRF represents our integreted CRF model. It demonstrates that Word Feature (WF) , Transition Feature (TF), Semantic Feature (SyF) and Syntactic Feature (SyF) contribute a lot to slot filling, while Start Feature (StF) and End Feature (EF) are less important. Also we can observe that the F1-score of our

[§]http://www.jfh.com

TABLE III: Slot filling results with different feature sets

Model	Precision	Recall	F1-score
Normal CRF	0.78	0.71	0.74
iCRF+WF	0.53	0.45	0.49
iCRF+TF	0.38	0.30	0.33
iCRF+EF	0.08	0.04	0.05
iCRF+StF	0.05	0.02	0.03
iCRF+SeF	0.30	0.28	0.29
iCRF+SyF	0.35	0.31	0.33
iCRF+WF+TF	0.63	0.57	0.60
iCRF+WF+TF+EF	0.65	0.58	0.61
iCRF+WF+TF+EF+StF	0.66	0.60	0.63
iCRF+WF+TF+EF+StF+SeF	0.75	0.68	0.71
iCRF+WF+TF+EF+StF+SeF+SyF	0.86	0.78	0.82

TABLE IV: Results of entity name unification

With Entity Name unification	Precision	Recall	F1-score
Yes	0.86	0.78	0.82
No	0.80	0.75	0.77

iCRF model reaches 82%, 8% higher than that of the normal CRF model.

Besides, in order to evaluate the effectiveness of entity name unification using SSKB, we performed slot filling with and without the preprocessing step. The results are shown in Table IV, which indicate that entity name unification preprocessing improves the performance of our integrated CRF method on the slot filling problem. Obviously, the number of words that are out of vocabulary can be significantly reduced through this preprocessing step.

V. Conclusion

This paper proposed an approach to building a task-oriented conversational bot (CrowDevBot) for software crowdsourcing platform. Several experiments are conducted to evaluate our approach and CrowDevBot, using the real data from JointForce. Experimental results show that the F1-score of our intention detecting mixture method is 87% under the limited training data, and the F1-score of our integrated CRF model with novel features to fill slots is 82%, up to 8 points higher than normal CRF. Also the user satisfaction ratio of CrowDevBot reaches 87% in average, which indicates that CrowDevBot really helps online users to finish software crowdsourcing tasks in real situation.

As for future work, we will leverage the reinforcement learning technology to improve the performance of our intention detecting and slot filling approach continuously, with better use of the user feedbacks.

Acknowledgement

This research was sponsored by the National Key Research and Development Program of China (Project No. 2018YFB1003903), National Nature Science Foundation of China (Grant No. 61472242 and 61572312), and Shanghai Municipal Commission of Economy and Informatization (No. 201701052).

References

[1] W. Wu, W. T. Tsai, and W. Li, "Creative software crowdsourcing: From components and algorithm development to project concept formations," *International Journal of Creative Computing*, vol. 1, no. 1, pp. 57–91, 2013.

[2] N. Gupta, G. Tur, D. Hakkani-Tur, S. Bangalore, G. Riccardi, and M. Gilbert, "The att spoken language understanding system," *IEEE Transactions on Audio Speech and Language Processing*, vol. 14, no. 1, pp. 213–222, 2006.

[3] R. De Mori, F. Bechet, D. Hakkani-Tur, and M. Mctear, "Spoken language understanding," *Signal Processing Magazine IEEE*, vol. 25, no. 3, pp. 50–58, 2008.

[4] V. Rieser and O. Lemon, *Natural language generation as planning under uncertainty for spoken dialogue systems*. Springer-Verlag, 2010.

[5] A. Mccallum and W. Li, "Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons," in *Proc. of the Conference on Computational Natural Language Learning*, 2003, pp. 188–191.

[6] J. D. Lafferty, A. Mccallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Eighteenth International Conference on Machine Learning*, 2001, pp. 282–289.

[7] F. Dernoncourt, Y. L. Ji, and P. Szolovits, "Neuroner: an easy-to-use program for named-entity recognition based on neural networks," 2017.

[8] M. Habibi, L. Weber, M. Neves, D. L. Wiegandt, and U. Leser, "Deep learning with word embeddings improves biomedical named entity recognition," *Bioinformatics*, vol. 33, no. 14, p. i37, 2017.

[9] T. H. Pham and P. Le-Hong, "End-to-end recurrent neural network models for vietnamese named entity recognition: Word-level vs. character-level," in *International Conference of the Pacific Association for Computational Linguistics*, 2017, pp. 219–232.

[10] Z. Yan, N. Duan, P. Chen, M. Zhou, J. Zhou, and Z. Li, "Building task-oriented dialogue systems for online shopping." in *AAAI*, 2017, pp. 4618–4626.

[11] M. A. Storey and A. Zagalsky, "Disrupting developer productivity one bot at a time," in *ACM Sigsoft International Symposium on Foundations of Software Engineering*, 2016, pp. 928–931.

[12] P. H. Su, M. Gasic, N. Mrki, L. M. R. Barahona, S. Ultes, D. Vandyke, T. H. Wen, and S. Young, "On-line active reward learning for policy optimisation in spoken dialogue systems," in *Meeting of the Association for Computational Linguistics*, 2016, pp. 2431–2441.

[13] T.-H. Wen, D. Vandyke, N. Mrksic, M. Gasic, L. M. Rojas-Barahona, P.-H. Su, S. Ultes, and S. Young, "A network-based end-to-end trainable task-oriented dialogue system," *arXiv preprint arXiv:1604.04562*, 2016.

[14] C. Lebeuf, M. A. Storey, and A. Zagalsky, "Software bots," *IEEE Software*, vol. 35, no. 1, pp. 18–23, 2018.

[15] Y. Kim, "Convolutional neural networks for sentence classification," *Eprint Arxiv*, 2014.

[16] P. Xu and R. Sarikaya, "Convolutional neural network based triangular crf for joint intent detection and slot filling," in *Automatic Speech Recognition and Understanding*, 2014, pp. 78–83.

[17] A. Jaech, L. Heck, and M. Ostendorf, "Domain adaptation of recurrent neural networks for natural language understanding," pp. 690–694, 2016.

[18] Y. Yang, W. Mo, B. Shen, and Y. Chen, "Cold-start developer recommendation in software crowdsourcing: A topic sampling approach." in *SEKE*, 2017, pp. 376–381.

[19] W. B. Cavnar, "N-gram based text categorization," in *Proc. Third Symposium on Document Analysis and Information Retrieval*, 1994, pp. 161–175.

[20] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *Computer Science*, 2013.

[21] Y. Y. Wang, A. Acero, J. Lee, and J. Lee, "Combining statistical and knowledge-based spoken language understanding in conditional models," in *Coling/acl on Main Conference Poster Sessions*, 2006, pp. 882–889.

[22] X. Li, "Understanding the semantic structure of noun phrase queries," in *Meeting of the Association for Computational Linguistics*, 2010, pp. 1337–1345.