Rebot: An Automatic Multi-modal Requirements Review Bot

1st Ming Ye
Operating System Product Dept.
ZTE Corporation
Chengdu, China
yeming@zte.com.cn

2nd Jicheng Cao
Operating System Product Dept.
ZTE Corporation
Chengdu, China
cao.jicheng1@zte.com.cn

3rd Shengyu Cheng✉
Operating System Product Dept.
ZTE Corporation
Chengdu, China
cheng.shengyu@zte.com.cn

***Abstract*— Requirements review is the process that reviewers read documents, make suggestions, and help improve the quality of requirements, which is a major factor that contributes to the success or failure of software. However, manually reviewing is a time-consuming and challenging task that requires high domain knowledge and expertise. To address the problem, we developed a requirements review tool, called Rebot, which automates the requirements parsing, quality classification, and suggestions generation. The core of Rebot is a neural network-based quality model which fuses multi-modal information (visual and textual information) of requirements documents to classify their quality levels (high, medium, low). The model is trained and evaluated on a real industrial requirements documents dataset which is collected from ZTE corporation. The experiments show the model achieves 81.3% accuracy in classifying the quality into three levels. To further validate Rebot, we deployed it in a live software development project. We evaluated the correctness, usefulness, and feasibility of Rebot by conducting a questionnaire with the users. Around 76.5% of Rebot's users believe Rebot can support requirements review by providing reliable quality classification results with revision suggestions. Furthermore, Around 88% of the users believe Rebot helps reduce the workload of reviewers and increase the development efficiency.**

***Index Terms*—Requirements Engineering, Requirements Review, Quality Model, Neural Network, Multi-modal Information**

I. INTRODUCTION

Requirements Engineering (RE) is an important part of software engineering that deals with the foremost stage in the software development. Extensive works show the software defects originating in deficient requirement analysis can cost more in a later phase [1], [2]. Therefore, it is important to perform quality controls since the very beginning of the process [3]. Requirements review has been an essential practice of quality assurance which helps to improve the quality of requirements and reduce potential defects. The current process of requirements review in ZTE corporation (a Chinese technology company specializes in telecommunication) requires reviewers with high domain knowledge and expertise to read and understand each requirement document, which is time-consuming and cumbersome. The requirements document here refers to the analysis of a single requirement which contains a series of necessary points about the requirement, such as background, application scenarios, acceptance criteria, etc. One option to improve the efficiency of requirements review is developing tools to automate the process.

To deploy any requirements review tools in ZTE, it must meet the following properties: (1) The tool must support parsing requirements documents in the form of web pages. (2) The tool is able to classify the quality of requirements automatically. (3) The tool can provide suggestions for modification to guide writers to quickly improve the quality of requirements. (4) The tool must support the Chinese language because the requirements are written in Chinese.

To our best knowledge, there is currently no tool with all of the above properties. Besides, existing works have been mostly restricted to textual patterns of requirements. Actually, the cognitive process that reviewers review a requirement document is from the surface to the inside. When a reviewer reads a requirement, the first thing that the reviewer notices is the layout appearance, which not only reflects the patterns of diagrams, tables, and text distribution but also indicates the level of requirement content organization. Nice layout organizations and rich presentational forms can make the requirement document more readable and understandable.

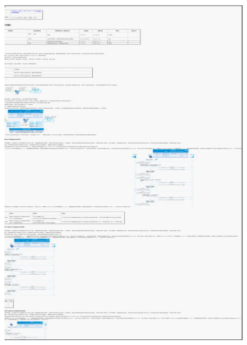
Fig.1 illustrates the intuition, we can tell the requirement document in Fig.1a is likely to be of higher quality than that in Fig.1b even without reading the text, as it has detailed tables, various diagrams, and a nice layout.

Visual layout is an important factor but not a decisive factor. The quality of the requirements document is a comprehensive consideration of text content and visual layout. Inspired by this intuition, we proposed a neural network-based quality classification model that combines the visual rendering with the textual content of requirements documents. Then an automatic tool has been developed based on the classification model.

In this paper, we report an industrial case study on developing, deploying, using, and evaluating a requirements review tool called Rebot, which automates the requirements parsing, quality classification, and suggestions generation.

The main contributions of this paper are as follows:

1) A neural network-based multi-modal quality classification model that takes visual and textual quality factors of requirements into consideration.


(a) High-quality requirement

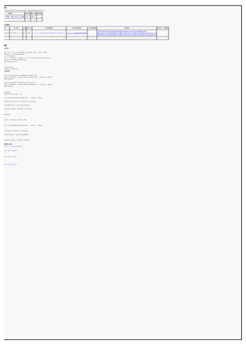

(b) Low-quality requirement

Fig. 1. Visual renderings of two example requirement documents with different quality labels (not intended to be readable).

2) An automatic end-to-end requirements review tool, which only needs the URL of requirements web page to provide the corresponding review results.
3) An empirical evaluation of Rebot in a live large-scale industrial environment.

II. RELATED WORK

With the complexity of requirements review, many efforts have been made to automate this review process. In the past twenty years, around 50 tools have been developed, but most of them do not work well due to their low accuracy and complexity [4]. Up to now, only a few tools are adopted by the industry, such as QVscribe [5], QuARS [6] and RQA [7]. QVscribe is an automatic tool that uses NLP technology to analyze the requirements at run time and assess their quality. QVscribe can provide a quality score for a requirement document based on some quality factors, such as immeasurable quantification, data-driven indicators, and vague words. QuARS is also an NLP tool for addressing interpretation problems at linguistic level. The core of QuARS is a quality model, which consists of quality properties to be evaluated through quality indicators. It is common for these tools to assess the quality of requirements by computing different quantitative metrics and integrating them. However, some researchers think the combination of quality metrics are rigid. Specifically, a linear combination of quality metrics is not sufficient to calculate a global measure of requirements quality. Parra et al. [8] applied machine learning algorithms to construct a quality classifier, which combines a group of quality metrics values extracted by RQA. More related works can be obtained through this review paper [4].

III. RESEARCH DESIGN

Fig.2 outlines the design of our research, which includes three main parts: the Quality Factors Identification, the Model Construction, and the Tool Building.

A. Quality Factors Identification

Our multi-modal tool fuses the visual and textual quality factors. As for the visual factors, we used a neural network to

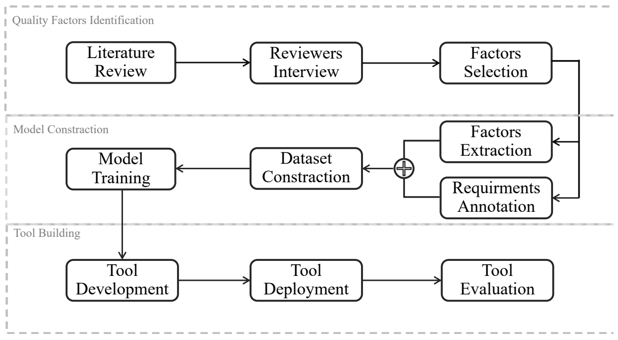


Fig. 2. Research design

capture the implicit visual quality factors from the screenshot of the requirements web pages. As for the textual factors, we firstly conducted a literature review to collect the widely used quality factors in the field. To complement the factors in literature, we interviewed requirements reviewers from multiple industrial projects of ZTE to collect the quality factors. Finally, we selected a subset of the identified quality factors. The selected factors must satisfy all of the following criteria: (1) The factor is mentioned in the existing literature. (2)The factor is recognized by the interviewers (3) The factor can be extracted by an automatic method. In other words, the quantitative data of factor can be automatically calculated by a program.

B. Model Construction

The core of Rebot is a neural network-based quality model, which uses quality factors to classify the requirements documents. For training this model, we constructed a real industrial requirements documents dataset with quality level labels (high, medium, low). We invited seven requirements reviewers with extensive professional experience to perform quality annotation. Besides, we extracted quality factors from requirements documents by an automatic parser. After the dataset is built, we trained and tuned the model until it reached satisfactory accuracy.

C. Tool Building

Based on the quality classification model, we developed a requirements review tool called Rebot. Then we deployed it in a live industrial environment and evaluated it by conducting a questionnaire about its correctness, usefulness, and feasibility.

IV. TOOL OVERVIEW

The architecture of Rebot is shown in Fig.3. There are five modules: the Front-End, the Parser, and the Quality Classification Model, the Database, and the Suggestion Generator.

A. Front-End

The Front-End takes the URL of the target requirements web page as input, and passes it to the Parser. After the review process is completed, the Front-End obtains the requirements

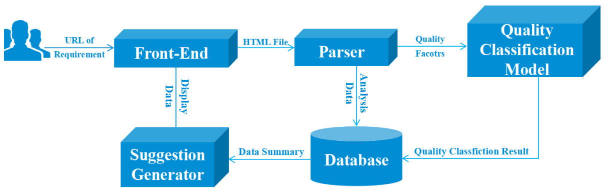
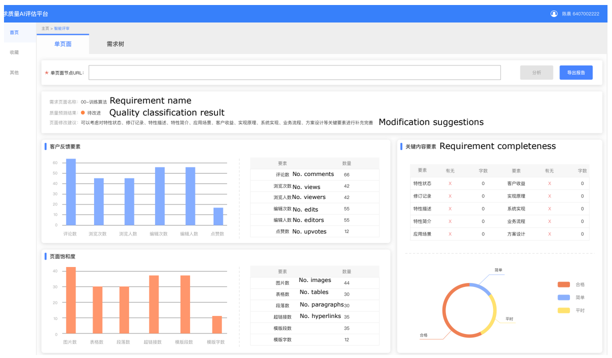


Fig. 3. The architecture of Rebot

quality classification result, some statistics of the page, and improvement suggestions from the Suggestion Generator. An example is shown in Fig.4.



需求页面名称: 00-训练算法 Requirement name
质量预测结果: ● 待改进 Quality classification result
页面修改建议: 可以考虑对特性状态、修订记录、特性描述、特性简介、应用场景、客户收益、实现原理、系统实现、业务流程、方案设计等关键要素进行补充完善 Modification suggestions

客户反馈要素

要素	数量
评论数 No. comments	66
浏览次数 No. views	42
浏览人数 No. viewers	42
编辑次数 No. edits	55
编辑人数 No. editors	55
点赞数 No. upvotes	12

关键内容要素 Requirement completeness

要素	有无	字数	要素	有无	字数
特性状态	X	0	客户收益	X	0
修订记录	X	0	实现原理	X	0
特性描述	X	0	系统实现	X	0
特性简介	X	0	业务流程	X	0
应用场景	X	0	方案设计	X	0

页面饱和度

要素	数量
图片数 No. images	44
表格数 No. tables	30
段落数 No. paragraphs	30
超链接数 No. hyperlinks	35
模版段数	35
模版字数	12

Fig. 4. The web user interface of Rebot

B. Parser

The Parser converts the target requirements web page into a 1000*2000 pixel screenshot, extracts some textual quality factors (see details in Table 1), and statistical indicators (number of views, viewers, editors, edits, comments, upvotes, etc.). All of the extracted data will be stored in the Database, while the screenshot and quality factors are sent to the Quality Classification Model. We remove the irrelevant information with requirements quality from the screenshot, such as the logo of ZTE, timestamps.

C. Quality Classification Model

The Quality Classification Model is a neural network model, which has been studied in our previous work [9]. We trained the model based on a industrial requirements dataset and store the best performance model in a pickle file. We used Flask framework to deploy the model and provide API accessing. Fig.5 demonstrates the architecture of the quality model, which consists of three components: the Visual Cognition Subnetwork, the Textual Characteristics Subnetwork, and the Concatenation Layer. The blue circles stand for visual features while the green ones stand for textual features.

1) Visual Cognition Subnetwork: To extract the visual quality factors of the requirements document (screenshot), we customized a fusion network based on the EfficientNets [10]. For capturing features at different scales, we parallelly connect EfficientNet-B2 with EfficientNet-B3, both are pre-trained on

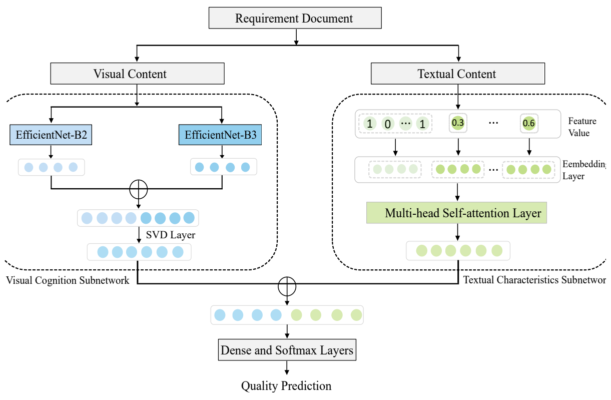


Fig. 5. The Architecture of Quality Classification Model .

ImageNet, as the backbone of Visual Cognition Subnetwork. Then we use the Singular Value Decomposition (SVD) layer to capture the most distinguishing feature vector as the output.

2) Textual Characteristics Subnetwork: This Subnetwork captures three types of quality factors of requirements. The first is completeness, which is measured by matching a requirement document template. The template is predefined to make sure the requirements documents containing all the necessary topics, for example, acceptance criteria. The second is the ambiguity of requirements, which is measured by the number of four types of specific words, e.g., vague, subjective, weak, and optional words. Thirdly, we extract some element statistics of the requirements documents, e.g. number of images. All the textual quality factors are shown in Table I. The "*" represents an arbitrary natural number. Now we have a group of features $F = \{f_1, f_2, ..., f_n\}$, where f_i is a boolean value or a scalar value. To allow the interactions between the two types of features, we firstly project them into the same feature space through an embedding layer. Then, we apply the multi-head self-attention layers [11] to model the correlation between different feature fields and get the output vector.

3) Concatenation Layer: We directly concatenate the output vectors respectively obtained from the two subnetworks. Then we apply a dense layer and a softmax layer to get the predicted result.

D. Suggestion Generator

Once all the quality factors have been calculated, their values are sent to the Suggestion Generator which is a rule-based decision model. A customizable configuration file is used to decide which rules are applied and what decision threshold is. The current generation rules are simple, i.e., if a certain factor triggers the specified condition, a corresponding reminder message is generated. For example, If the word count of a topic is lower than the threshold, the suggestion will be: "Please enrich the content of *[name of topics]*".

V. EVALUATION

In this section, we present the dataset construction and the results of Rebot's evaluation, which contains the predictive

TABLE I
TEXTUAL QUALITY FACTORS.

Quality Factor	Value	Notes
Completeness	0/1	Scenes, Principle, Background, Test Scheme, Release Version, Revision Record, Reliability Tests, Acceptance Criteria, Interface Parameters, Characteristic Status
Ambiguity	*	No. (Number of) Vague words (e.g. recent, adequate,...), subjective words (e.g. efficient, simple,...), weak words (e.g. could, may,...), optional words (e.g. possibly,...)
Elements Statistics	*	No. Tables, No. Images, No. Hyperlinks, No. Paragraphs, No. Words for each topic, No. Images for each topic, No. Revision, No. blanks in particular tables

performance evaluation of the quality classification model and the questionnaire evaluation of Rebot.

A. Dataset Construction

To train and evaluate the Quality Classification Model, we constructs a real industrial requirements dataset by collecting requirements documents from ZTE. Seven requirements reviewers are invited to perform quality annotation (one document has a quality label). Each requirement is randomly assigned to three reviewers. To ensure the quality of the labels, the reviewers abide by the following criteria: (1) *Correctness:* Whether the description of the requirement is consistent with the goal of the product. (2) *Unambiguity:* Whether the description is clear and unambiguous. (3) *Scenarios:* Whether the application scenarios are described completely. (4) *Interfaces:* Whether the user interface and interfaces between modules are defined. (5) *Acceptance:* Whether the specifications of function, performance, safety, and reliability are included. A requirement is accepted as a valid sample if the corresponding three reviewers give the same labels. Otherwise, it will undergo a new round of labeling. If the results are still inconsistent, the three reviewers will discuss its quality together to give a final label. Finally, we got 628 raw requirements documents with quality labels (246 high, 205 medium, and 177 low). We randomly select 128 samples as a test set and enlarge the remaining 500 samples by six times with image data augmentation techniques for training and validating. The data set is expanded by image augmentation, so that over-fitting is mitigated when training the neural network model, and a better model can be obtained in the end. Detailed statistics of our dataset are shown in Table II.

TABLE II
DATASET STATISTICS.

Dataset	High-quality	Medium-quality	Low-quality	Total
Training	1071	855	774	2700
Validation	119	95	86	300
Test	50	42	36	128

B. Quality Classification Model Evaluation

We train the model on the above dataset. Parameters are set as follows. In the visual cognition subnetwork, the pre-trained Efficientnet-B2 and Efficientnet-B3 keep the default settings and produce a 2,688 dimension vector. The SVD layer is set to obtain the most significant 256 features. In the textual characteristics subnetwork, the embedding layer has a size of 64 and then we use a multi-head self-attention layer with 2 heads and a hidden size of 32 for each head. Besides, we adopt Adam optimizer with a learning rate of 0.0001 and a batch size of 16. To prevent overfitting, we adopt the early stop training strategy. We optimize our network based on the cross-entropy loss for classification.

In order to respectively explore the impact of visual and textual factors on quality assessment, we construct Model-V based on the Quality Classification Model with only enabling the visual cognition subnetwork and Model-T with only enabling the textual characteristics subnetwork. The results in Table III show that the accuracy of Model-VT achieves 81.3%, which is higher than that of Model-V and Model-T. This validates the effectiveness of Quality Model and the complementarity of visual information and textual information. Besides, the recall for the low and medium quality requirement documents achieved 91.7% and 73.8% respectively, thus enabling most of the requirements documents that need to be improved can be identified.

TABLE III
PRECISION ("P"), RECALL ("R"), AND ACCURACY ("A") OF THE QUALITY CLASSIFICATION MODEL AND ITS VARIANTS ACROSS THREE LEVELS.

Quality	Metrics	Model-T	Model-V	Model-VT
High	P	71.1%	80.0%	**85.1%**
	R	64.0%	72.0%	**80.0%**
Medium	P	58.1%	60.4%	**72.1%**
	R	59.5%	69.0%	**73.8%**
Low	P	65.0%	80.0%	**86.8%**
	R	72.0%	77.8%	**91.7%**
All	A	64.8%	72.7%	**81.3%**

C. Tool Evaluation

To evaluate the correctness, usefulness, and feasibility of Rebot in a real industrial environment, we first deployed it to a project of ZTE. The business analyzer (BA) can use it to self-check their requirement documents and the reviewers can use it to aid the review process. Then we conducted a

TABLE IV
USERS' PERCEPTIONS REGARDING REBOT

Question	Strongly agree	Agree	Neutral	Disagree	Strongly disagree
Rebot is easy to understand and use	23.6%	70.6%	5.9%	0%	0%
Rebot's quality classification results are reliable	23.6%	47.1%	17.6%	11.8%	0%
Statistics provided by Rebot is useful	35.3%	47.1%	11.8%	5.9%	0%
Suggestions provided by Rebot are useful	5.9%	41.2%	23.6%	23.6%	5.9%
Rebot helps reduce the workload of reviewers	70.6%	17.6%	11.8%	0%	0%

questionnaire with 17 users who have used it for more than two weeks. The questionnaire consists of five multiple-choice questions and one subjective question about Rebot's views. The statistics for the feedback data on these questions are shown in Table IV.

As a result, all of the users either agree (70.6% strongly agree and 17.6% agree) or are neutral, and nobody disagrees that Rebot can help reduce the workload of reviewers. Rebot filter out low-quality requirements in advance, allowing reviewers to focus on high-quality requirements, thereby greatly reducing the workload. Meanwhile, 94.2% of the users agree (23.6% strongly agree and 70.6 agree) and 5.9% are neutral that Rebot is easy to understand and use. Furthermore, the majority of the users agree that the quality classification results and statistics provided by Rebot are reliable and useful. However, 47.1% of the users agree and 29.5% disagree (23.6% disagree and 5.9% strongly disagree) that the suggestions generated by Rebot are useful. Through the feedback on the subjective question, we learned that they think the suggestions are too simple to effectively guide the revision work. Fortunately, users can know the direction of improvement through the statistical data Rebot provides.

VI. THREATS TO VALIDITY

Rebot does have some limitations in general. Primarily, the quality model does not perform semantic-level analysis of textual content. This may cause some requirements documents with good visual layout but poor content description to be classified as high quality or poor visual layout but good content description to be classified as low quality.

Secondly, regarding internal validity, one limitation is that Rebot has only considered a subset of textual quality factors. There are many quality factors of requirements that can continue to be studied, such as singularity, quantification, and accuracy. Thirdly, in relation to external validity, our research has limited generalizability: we have deployed and evaluated Rebot in just one case (one project and one company). Finally, one of the reliability limitations of this research is that we cannot open the data used in this investigation to the research community due to the company policy.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we conducted a case study that results in a requirements reviews tool, called Rebot, which automates the requirements parsing, quality classification, and suggestion generation. We introduced neural network to integrate multi-modal quality factors for constructing a quality classification model. We have deployed Rebot in a real large-scale industrial environment and evaluated its correctness, usefulness, and feasibility by a questionnaire. The overall conclusion is that Rebot is easy to use and can provide reliable review results, in turn, helps reduce the workload of the reviewers along with decreasing the overall time of requirements review.

While the results indicate it is worth using Rebot to help with requirements review, there are still many points to be improved. We plan to first quantify more quality factors related to requirements descriptions, such as singularity, quantification, and accuracy. Then we plan to collect more requirements documents to expand the data set to train a more powerful classification model. Besides, the questionnaires are not enough to really understand how the users perceive Rebot. we plan to do complementary interviews in the future. Finally, deploy Rebot to more projects and continue to evaluate its performance.

REFERENCES

[1] Henning Femmer. Reviewing natural language requirements with requirements smells:a research proposal. In *International Doctoral Symposium on Empirical Software Engineering*, 2013.

[2] IEEE Computer Society. Swebok guide to the software engineering body of knowledge. http://www.computer.org/portal/web/swebok, 2015. The Standish.

[3] R. Zeist and P. Hendriks. Specifying software quality with the extended iso model. *Software Quality Journal*, 5:273–284, 12 1996.

[4] Afrah Naeem, Zeeshan Aslam, and Munam Shah. Analyzing quality of software requirements; a comparison study on nlp tools. pages 1–6, 09 2019.

[5] Matthew Cooper Olivia Kenney. Automating requirement quality standards with qvscribe. https://qracorp.com/qvscribe/, 2019.

[6] Stefania Gnesi, Giuseppe Lami, and Gianluca Trentanni. An automatic tool for the analysis of natural language requirements. *Computer Systems Science and Engineering*, 20, 01 2005.

[7] Gonzalo Genova, Jose M. Fuentes, Juan Llorens, Omar Hurtado, and Valentin Moreno. A framework to measure and improve the quality of textual requirements. *Requirements Engineering*, 18, 03 2011.

[8] Eugenio Parra, Christos Dimou, Valentin Moreno, Juan Llorens, and Anabel Fraga. A methodology for the classification of quality of requirements using machine learning techniques. *Information and Software Technology*, 67, 07 2015.

[9] Ming Ye, Jicheng Cao, Shengyu Cheng, Dong Liu, Shenghai Xu, and Jinning He. Mrdqa: A deep multimodal requirement document quality analyzer. *2021 IEEE 29th International Requirements Engineering Conference (RE)*, pages 446–447, 07 2021.

[10] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *International Conference on Machine Learning(ICML)*, abs/1905.11946, 2019.

[11] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. Autoint: Automatic feature interaction learning via self-attentive neural networks. pages 1161–1170, 11 2019.