A Preliminary Framework for Recognizing iStar Hand Drafts

Yuran Zhu and Tong Li*

Beijing University of Technology,
100 Ping Le Yuan, Beijing 100124, China
{zhuyuran@emails.bjut.edu.cn, litong@bjut.edu.cn}

Abstract. Despite the rapid development of iStar modeling tools, drafting iStar models on paper is convenient for illustration purposes and can be inspiring during brainstorming sessions. Digitizing such hand drafts and generating corresponding model files can enable modelers to save and reuse model snippets. This paper presents a preliminary technical framework for achieving this goal. Adopting the divide and conquer strategy, the proposed framework consists of three separate recognition modules dealing with node recognition, text recognition, and element composition. Upon implementing the framework, the recognized model drafts will be transformed into iStarML files to be easily imported to iStar modeling tools. This paper focuses on discussing the feasibility of the proposed approach and presents some preliminary.

Keywords: iStar model, Hand draft, Model recognition.

1 Introduction

Despite the rapid development of iStar modeling tools [1,2], drafting iStar models on paper is still popular given the convenience of producing illustrations by hand. However, hand drafts of iStar models are difficult to archive and reuse. One solution to this problem can be digitizing such hand drafts. The major challenge in implementing this solution is the lack of methods for precise recognition of the graphic iStar notations and assembling them into an iStar model.

One existing relevant method uses a support vector machine (SVM) to recognize nodes [3]. However, SVMs do not support multi-object detection. Object detection using deep learning provides another option [4]. Given the similarity between the node and object detection tasks, it is possible to apply object detection methods to recognize nodes in iStar hand drafts.

This paper presents a preliminary framework for automatically recognizing iStar hand drafts and generating corresponding iStar model files while preserving the position

* Corresponding author

information of all drafted elements. In particular, the paper reports on our ongoing research on the text recognition and element composition modules with an emphasis on discussing the feasibility, advantages, and disadvantages of our proposal.

2 Proposal

The proposed framework for transforming iStar hand drafts to iStarML files comprises six modules: preprocessing, node recognition, edge recognition, text recognition, element composition, and outputting iStarML files. Some parts of the framework are simplified based on some features of the iStar 2.0 model such as the arrangement in position of parent and child nodes. The framework takes original images as input and outputs corresponding iStarML files. Fig. 1 illustrates the modules and data flows within the framework.

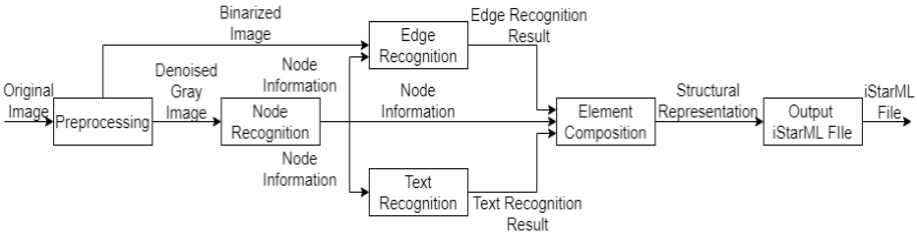


Fig. 1. Modules and Data Flows within the Proposed Framework

2.1 Preprocessing

The preprocessing module takes an original image as input and outputs the corresponding binarized and grayscale processed images. This module first converts three-channel RGB images to one-channel original grayscale images. After that, image denoising algorithms such as NL-Means Algorithm [5] would be applied to the original grayscale images to remove noise and output denoised grayscale images. Binarized images would be generated based on the denoised grayscale images. For the convenience of applying edge recognition using Breadth First Search (BFS), the denoised grayscale images should be transformed to matrices of 0s and 1s, where 0 represents “white” and 1 represents “black.” Any pixel under an empirical threshold selected manually would be set to 0; otherwise, pixels would be set to 1. Resizing the denoised grayscale images is not necessary in the proposed framework given that it uses the YOLOv3 network [6] in the node recognition module. If another method is employed for node recognition, then resizing may become necessary.

2.2 Node Recognition

Deep learning was employed for node recognition. Compared to other deep learning methods, YOLOv3 has the advantages of low computational cost and high accuracy

[6]. Among other YOLOv3 network architectures, YOLOv3-tiny network was selected as it can be easily trained using a small amount of training data; it was also found to perform well enough in the node recognition task.

The node recognition module of the proposed framework takes the denoised grayscale image output by the preprocessing module as input and outputs node information comprising the category and position of the node. Fig. 2 shows the preliminary node recognition result generated by the YOLOv3-tiny network. While the preliminary result was obtained using a software draft rather than a hand draft, the network is expected to perform well in recognizing hand drafts after collecting enough hand-draft samples for training the network.

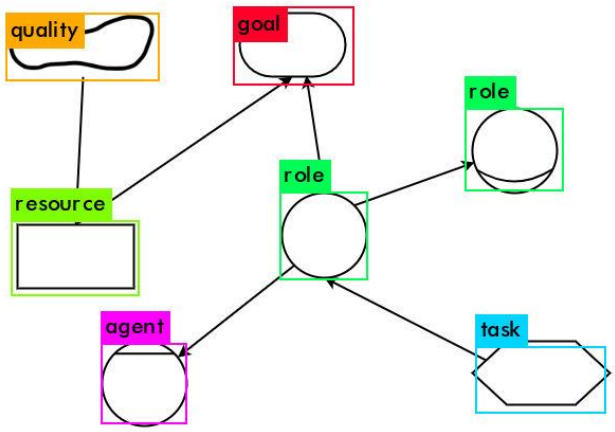


Fig. 2. Preliminary Node Recognition Result

In the future implementation, elements containing the information of edges, such as whether the arrow is present at the tail of an edge, will be added to the node recognition module. While the module will not recognize edges as objects, it will indicate their types.

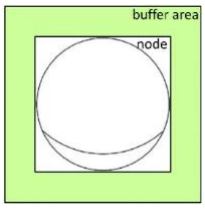
2.3 Text Recognition

While text recognition is a well-developed area [7-8], associating recognized text with iStar nodes is still a remaining task. In the proposed framework, node information and its denoised grayscale image are fed into the text recognition module. A text recognition method would be applied only to the pixels within the border of an iStar node; then, the result of text recognition would be associated with the specific iStar node that shares the same area in the image. The combination of a convolutional neural network (CNN), Seq2Seq, and an attention network is selected as the text recognition method [9].

2.4 Edge Recognition

In the preliminary framework, all edges in an image are assumed to be solid lines. The basic idea of solving the edge recognition problem is that one node is associated with another node if the head and tail of an edge are close enough to these nodes separately. To quantify the notion of "close enough," a concept called buffer area is introduced. The buffer area is a rectangular area around a node, which indicates the potential area where an edge starts or ends. Fig. 3 shows the buffer area around a node. The edge recognition module would generate edge recognition result comprising a pair of nodes associated together as output.

If the head and tail of an edge fall into the buffer area of two nodes separately, then these two nodes would be associated together. After detecting all nodes in an image, the edge recognition module would retrieve the position of all the nodes from node information generated by the node recognition module. The edge recognition module would iterate through all the nodes sequentially. For each node, all its pixels would be scanned through. If any pixel is 1, it would be fed as the start of the BFS process. The BFS method would search through all the adjacent black pixels set to 1. If any black pixel falls into any buffer area, then the node that buffer area belongs to would be associated with the node containing the pixel used as the start of the BFS process. Fig. 3. Buffer Area around a Node **Fig. 4.** Process of Associating Two Nodes illustrates the process of associating two nodes.



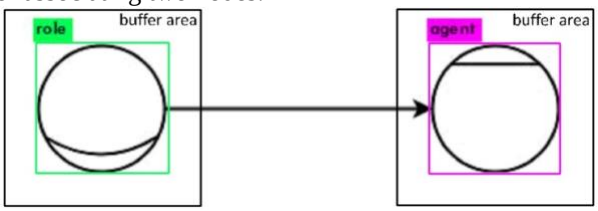


Fig. 3. Buffer Area around a Node **Fig. 4.** Process of Associating Two Nodes

Considering that pictures of hand drafts can be taken from different distances and angles, the buffer area may appear on a different scale, leading to different results from the same image. To solve this problem, the ratio of the node scale is set to be the criterion for determining the scale of the buffer area.

During the BFS process, if an edge overlaps with the area of any node containing the element representing edge types, then the framework would mark that edge as being of the found type. Considering the rules of refinement, edges pointing to one specific parent node should be of the same type. While the edge recognition module may fail to recognize and mark some edges correctly, the framework would still be able to mark the remaining edges pointing to a specific parent node correctly, as long as the type of any one edge pointing to that parent node is determined, which makes the proposed framework robust.

To recognize the type of each edge, it is necessary to identify the parent node of a pair of nodes. To simplify this task, a parent node can be considered as a node above a child node. The framework exploits the positions of nodes output by the node recognition module to determine whether a node is s parent or child node in a pair of nodes.

2.5 Element Composition & Outputting iStarML File

The element composition module would take the edge recognition result, node information, and text recognition result as input, and output the structural representation of the identified elements. Due to the modular architecture on the proposed framework, the results necessary for generating a final iStarML file do not share a uniform representation. In the element composition module, the edge recognition result, node information, and text recognition result would be merged to a uniform and structural representation of all elements representing the entire iStar model structure in the original image. The output iStarML file module would take the structural representation of all elements as input and output an iStarML file, which could be easily imported to iStar modeling tools.

3 Related Work

There are several related studies aiming at solving the problem of transforming iStar hand drafts into machine-readable model files, e.g., iStarML files. For example, Gilbert and Rusli [10] applied a faster R-CNN network to detect separate elements in hand-drafted iStar models; however, the authors did not address the relationship recognition task and thus cannot assemble separate elements to form a complete digital iStar model. Wu and Zhang used an SVM to classify nodes of a hand draft [3]; however, traditional SVMs do not support multi-object detection and thus require grouping the elements of an input image in advance.

Shi and Luo [11] employed Random Forest to classify nodes according to their types but not to construct complete models out of them. Feng and Zhao used the Sketch-semantic Net, which combines AlexNet and WordNet, to transform model hand drafts to machine-readable files [12]. This approach relies on the semantic information within nodes to determine the relationship between nodes. However, the semantic information does not always reflect the relationship explicitly.

4 Conclusion & Discussion

This paper presented a preliminary framework for recognizing iStar hand drafts and transforming them to iStarML files. The proposed framework consists of six modules, namely, preprocessing, node recognition, edge recognition, text recognition, element composition, and outputting iStarML files. We, in this paper, focus on discussing the

feasibility, advantages, and disadvantages of the methods employed in the proposed framework, shedding light on plausible improvements in the future. As for our next step work, for one thing, we plan to develop a prototype tool that implements our proposed approach; for another thing, we aim to conduct case studies with iStar modelers and iteratively improve our approach based on their feedback.

Acknowledgment

This work is partially supported by the National Natural Science of Foundation of China (No.61902010), Beijing Excellent Talent Funding-Youth Project (No.2018000020124G039), and Engineering Research Center of Intelligent Perception and Autonomous Control, Ministry of Education.

References

1. Jennifer Horkoff, Neil A. M. Maiden. (2015). Creative Leaf: A creative iStar modeling tool. *2015 iStar*, pp. 25-30.
2. Yiwen Chen, Yuanpeng Wang, Yixuan Hou, Yunduo Wang. (2019). T-Star: A Text-Based iStar Modeling Tool. *2019 IEEE International Conference on Requirements Engineering*, pp. 490-491.
3. Jie Wu, Liqing Zhang. (2014). Sketch recognition with natural correction and editing, pp. 951-957.
4. Zhong-Qiu Zhao, Peng Zheng, Shou-Tao Xu, Xindong Wu. (2019). Object detection with deep learning: A review. *2019 IEEE Transactions on Neural Networks*, vol. 30, issue. 11, pp. 3212-3232.
5. A. Buades, B. Coll, J.-M. Morel. (2005). A non-local algorithm for image denoising. *2005 Computer Vision and Pattern Recognition*, vol. 2, issue. 2, pp. 60-65.
6. Joseph Redmon, Ali Farhadi. (2018). YOLOv3: An Incremental Improvement. *2018 arXiv: Computer Vision and Pattern Recognition*.
7. D Ghosh, T Dube, A P Shivaprasad. (2010). Script Recognition-A Review. *2010 IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, issue. 12, pp. 2142-2161.
8. Pratik Madhukar Manwatkar, Kavita R. Singh. (2015). A technical review on text recognition from images. *2015 IEEE International Conference on Intelligent Systems and Control*, pp. 1-5.
9. Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, et al. (2018). DiSAN: Directional Self-Attention Network for RNN/CNN-Free Language Understanding. *2018 National Conference on Artificial Intelligence*, pp.5446-5455.
10. Nathanael Gilbert, Andre Rusli. (2020). Single object detection to support requirements modeling using faster R-CNN. *2020 TELKOMNIKA Telecommunication Computing Electronics and Control*, vol. 18, issue. 2, pp. 830-838.
11. Dapeng Shi, Bin Luo (2016). Research in Domain-oriented Sketch Recognition Method. Nanjing University.
12. Chencheng Feng, Peng Zhao (2019). The Research of Sketch Recognition Combining with Semantic Information. Anhui University.