Activities and Findings - Summmer 2009

Joshua Ehrlich

July 24, 2009

1 Summary of Activities

This summer our focus was on refining the recognition process. I repaired CircuitSimulatorUI, which had been written in the summer of 2007. CircuitSimulatorUI is designed to be a complete system that includes a sketch panel for the user to draw sketches and then it classify the single strokes, groups the strokes, and recognizes the groups. It also will form semantic meaning for the circuit. I removed the old recognition tool, which did not have reliable results, and integrated Eric Petereson's more sophisticated one. I also repaired some of the labeler functionality during this time to allow the user to correct errors made during recognition.

I also worked on CircuitRec, a domain specific project designed to provide the semantic meaning for CircuitSimulatorUI. CircuitRec recognized wires and labels and provided feedback about the way the circuit was assembled. Originally CircuitRec was only designed to handle perfectly recognized sketches, which meant that the user had to correct all of the the recognition errors with the labeler before continuing. I began by removing this constraint allowing the CircuitRec to take imperfect sketches and updating the feedback it provides to indicate possible errors in recognition. I tested this with the TestCircuitRec project and made sure that it performance was unchanged from the original on the correct sketches. I reworked the wire recognition to fix an error that was incorrectly grouping crossing wires. This error was due to the original wire connection algorithm considering every endpoint of every substroke when trying to make connections even though many of these endpoints are not actually endpoints of the wires.

After Denis returned, we met with Eric and decided to focus on a systematic refinement process that would start with a blank sketch and by iteration eventually recognize the entire circuit. After deliberation we decided to implement simultaneous recognition for each class. To accomplish this, I refactored CircuitRec to extract the wire recognizer and text recognizer. We wanted to have uniform shape recognition, so we created the ShapeRecognitionResult as the standard of what shape recognizers should return. Unfortunately, while Eric's gate recognizer returns an actual probability, the text recognizer only returns one of three values, and the wire recognizer one of two. The reason for this

is that the text recognizer is a wrapper around the Microsoft text recognizer, which returns Strong, Medium, or Poor rather than a percentage. In addition, because wires are only characterized by how they are connected, the wire recognizer can only return a 1 if it is possible that the shape is a wire or a 0 if it is not possible that the shape is a wire. This created a problem when tyring to compare diiferent ShapeRecognitionResults.

The final part of this new refinement scheme that I worked on was integrating domain specific context. The first part of this was the creation with Denis of the ContextDomain. ContextDomain is designed to determine based on domain specific rules which shapes are connected and what relationships between shapes are valid. The second part was the creation of a Baysian classifier that combines the results of the different shape recognizers, the classifications of the substrokes, and the types of connected shapes to produce a probability that a label applies to the shape. I implemented a naive bayes classifier and after several tests settled on 3 features. The first feature is the context of the shape, a sorted list of the classes of the connected shapes. The second feature is the number of substrokes in the shape. The final feature is the combined result of the three shape recognizers.

2 Finidings

The naive Bayes classifier was trained on 160 files from the NewConnectedClassified folder. These files are sketches from the GateStudyData that were labeled with the correct connections and correct classifications. Because they are the labeled sketches from the GateStudyData, they also have the correct groups and labels. I tested the classifier on the remaining 40 files in the NewConnectedClassified folder [1].

Using the original feature set which included a continuous feature for the percentage of substrokes classified as each type and did not have the substroke count feature I found the following results. Note that these results only are from our original training and testing set which had errors in the connection code and used the output of the actual stroke classifier to classify single strokes.

		ACTUAL LABEL									
		AND	LABEL	LABELBOX	NAND	NOR	NOT	NOTBUBBLE	OR	WIRE	XOR
P	AND	.95	.12	0	.25	.06	.11	.62	.27	.01	.07
R	LABEL	0	.63	0	0	0	0	0	0	.01	0
E	LABELBOX	0	0	.88	0	0	0	0	0	0	0
D	NAND	0	.01	0	.63	.13	.02	0	.15	0	.03
I	NOR	0	.01	0	.06	.63	.06	0	.02	0	.07
C	NOT	0	.03	.03	0	.06	.76	0	.05	0	0
T	NOTBUBBLE	0	.09	0	0	0	0	.38	0	0	0
E	OR	.01	.01	0	.06	0	.02	0	.35	0	.03
D	WIRE	.04	.09	.09	0	0	0	0	.15	.98	.03
	XOR	0	.01	0	0	.13	.03	0	.02	0	.77

[1] these files are located in NewConnectedTesting

This recognizer outperforms the original CombinationRecognizer in every category except OR gates. In addition its Label recognition is lower than most of the other classifiers we experimented with. The issue being that the single-stroke feature gave AND gates significantly more weight than the other gates.

I experimented with modifying the different features and for more detailed results look at Comparison of Different Classifiers. I evaluated the classifier on the original test set for each combination of at least two features [2].

I retrained the classifier after realizing an error had occurred in the connections of notbubbles. This produced slightly different results

In general at this point, the best set of features appears to be using the priors, the connection feature, the number of substrokes, and the result of the gate recognizer only. With this feature set, even given errors in connections, we correctly recognize 97% of label boxes, 81% of and gates, 82% of not gates, 61% of or gates, 81% of xor gates, 37% of nor gates, 28% of nand gates and 83% of notbubbles. We also recognize 93% of labels and 88% of wires correctly. The places where this performs poorlt are all areas that the gate recognizer also performs poorly.

The confusion matrix for this final classifier is given below

		ACTUAL LABEL									
		AND	LABEL	LABELBOX	NAND	NOR	NOT	NOTBUBBLE	OR	WIRE	XOR
P	AND	.81	0	0	.22	0	.04	0	.28	0	0
R	LABEL	.09	.93	.03	0	0	.04	0	.07	.03	0
E	LABELBOX	0	0	.97	0	0	0	0	0	0	0
D	NAND	.02	0	0	.28	0	0	0	0	0	0
I	NOR	0	0	0	.06	.37	.01	0	0	.03	.06
C	NOT	0	.02	0	.06	.16	.82	0	0	.02	0
T	NOTBUBBLE	.02	.01	0	0	0	0	.83	0	.01	0
E	OR	.0	.01	0	.39	.21	.01	0	.61	.01	.13
D	WIRE	.02	.02	0	0	0	0	.17	.02	.88	0
	XOR	0	.02	0	0	.26	.01	0	.03	.02	.81

From the confusion matrix it becomes clear that the general issues arise mostly from gate recognition amongst similar classes. For instance or gates are misclassified as and gates 28

3 Future Research

Future research could be used in two areas. First the naive Bayes currently only uses 3 conceptual features: the combinde shape recognition feature, a connectivity feature, and a single-stroke classification feature, and these features could be modified and new features could be added to try to improve classification performance. In particular, the single-stroke classification feature is unreliable so we are currently not using it; however, based on our original testing it may

[2] counting the prior as a feature

be a useful feature in some cases. While the connection feature is usesful, work could be done to improve the connection info, allowing for more sophisticated features. Second the Bayes classifier needs to be integrated into the refinement process. Ideally the classifier would be tested on its ability to recognize partial shapes and its ability to distinguish between good shapes and bad shapes.