EUROGRAPHICS Symposium on Sketch-Based Interfaces and Modeling (2012)
L. B. Kara and K. Singh (Editors)

Automated Labeling of Ink Stroke Data

Jacky (Shunjie) Zhen, Rachel Blagojevic, Beryl Plimmer

Department of Computer Science
University of Auckland
Private Bag 92019, Auckland, New Zealand

Abstract

Labeled ink stroke data is essential to the development and evaluation of sketch recognizers. Manually labeling strokes is a tedious, time-consuming, and error prone task; and very few tools are available to facilitate this. We propose a new and intuitive method of automatic labeling for single stroke primitives. This involves building a recognizer from a partially labeled dataset. This recognizer is then used to identify and automatically label the remaining data, therefore reducing the amount of manual labeling required by researchers. An evaluation comparing manual labeling against our new auto labeling method shows that users are able to label significantly faster and produce less errors using auto labeling. Furthermore, users found auto labeling easier and more preferable.

Categories and Subject Descriptors (according to ACM CCS): I.7.5 [Document and Text Processing]: Document Capture: *Graphics recognition and interpretation*, I.2.5 [Artificial Intelligence]: Programming Languages and Software: *Expert system tools and techniques*.

1. Introduction

A central part of developing more accurate sketch recognizers is the collection and analysis of quality digital ink data. There is a growing desire to standardize and simplify the process of collecting ink data, in order to support the collaborative efforts and the corroboration of results. For example, SOUSA [PWJH08, KJM*09] is an online applet allowing researchers to collect and verify sketches. Users can participate in this study via the website and sketch the required diagrams. Although their web interface provides ease of access to search, collect, and download data; it does not provide any labeling facility. Labeled data is required to provide ground truth information for evaluation or testing purposes, and developing recognizers using supervised learning techniques.

A commonly identified problem is the difficulty of data labeling, largely due to the tedium and time requirements of manually labeling large sets of data. Previous work [WSA07, BPGW08] has attempted to reduce the effort of labeling by providing tools specifically tailored to manual labeling. However, labeling continues to present a challenge for sketch recognition researchers, as the amount of data in sketch datasets scale to greater numbers, increasing the labour requirements for manually labeling the data. Relief may be found in the automation of dataset labeling. Automation presents itself as a scalable solution that counters both the tedium and effort demanded for the labeling of ever increasing datasets.

We present, an innovative and intuitive way to automate the assigning of labels to stroke data, based on previous research into data mining for stroke recognizers [CBP12]. The technique involves the generation of an on-the-fly recognizer, created by modeling a small manually labeled subset of the dataset; the generated recognizer is used to automatically label the majority of the dataset. We implemented the technique by integrating the auto labeling function into the labeling component of RATA [CBP12], a complete software toolkit aimed at non-experts for generating single stroke gesture recognizers. Initially, using the RATA labeling component interface, the user manually labels some of the strokes within a dataset and then invokes the auto labeling function for the rest. The user then manually corrects errors made by the automated function. Figure 1 shows the RATA labeling interface.

The main contributions of this paper are: the development of a new auto labeling method, and an improved manual labeling experience. Both contributions are designed to remediate the increasing demands of the manual labeling process for sketch recognition research by reducing the time and effort required while still providing accurate labels.

DOI: 10.2312/SBM/SBM12/067-075

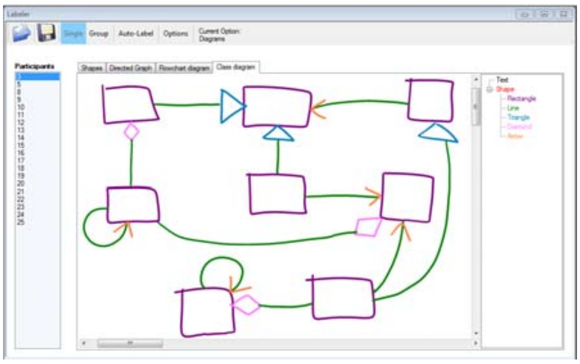

Figure 1: *The RATA labeling component interface implemented with auto labeling function access from top control tab.*

2. Related Work

Several tools have been developed to assist in labeling digital ink data. Wolin et al [WSA07] designed a tool for more efficient labeling of ink data using a stylus. Their tool is able to complete three main tasks; stroke fragmentation (automatic and manual), stroke grouping (manual), and symbol labeling (manual). Fragmentation derives ink primitives (such as lines and arcs) by either splitting strokes, or separating strokes that contain more than one symbol. Stroke grouping works in the opposite direction by joining components made of more than one stroke. Once these tasks have been performed, manually labeling the symbol in the sketch can be carried out. Their tool also allows for multiple labels to be applied to a stroke. The only automated process available in this application is for fragmentation of strokes.

DataManager [BPGW08, BSP09] is a tool designed for sketch data collection, labeling, and automatic dataset generation. The labeling scheme allows for multiple labels to be applied to a stroke, and group labeling (for components containing more than one stroke). There is also an inbuilt automatic labeler for classifying strokes into either shapes or text using a divider algorithm [BPGW08]. Any corrections that need to be made can then be done manually. This study showed that using DataManager greatly improved the efficiency of labeling. However, the automated labeler is limited to the broad classification of ink into either text or shape categories– different classes of shapes are not classified.

MacLean et al [MTL*09] developed a tool for collecting and labeling handwritten mathematical expressions. Their goal was to generate a corpus of data by limiting the bias resulting from researchers collecting only the type of data that their recognizers are more familiar with. To ensure their datasets are as varied as possible they use random walks of a grammar-based mathematical expressions recognizer to generate a large range of expressions for participants to transcribe. Participants write these expressions and strokes are then labeled automatically, using a recognition algorithm for mathematical expressions. Unfortunately, this tool is limited to the collection and labeling of mathematical expressions.

Several tools have been developed focusing more on sketch data collection. These include web-based games [Joh09, JD09], and an online applet, SOUSA [PWJH08, KJM*09]. These tools do not provide a labeling facility for individual parts of a sketch, but have user defined descriptions of the sketch as a whole.

Frameworks have also been developed for building and evaluating gesture recognizers, which include data collection and labeling functions [BLMM07, SKN07, SNK07, AFGP08, MMBK09]. These systems are typically used to collect isolated gestures or components rather than full diagrams.

One reason why researchers have preferred to collect isolated components is that labeling is easier - the label for a shape can simply be inferred by the diagram name, in contrast to full diagrams where each stroke must be labeled individually. Unfortunately, isolated components lack important spatial and temporal contextual information that may exist between gestures. Training with such data has been found to have a direct effect on recognition accuracy if the recognizer is to be applied on more complex sketches [FGP*09, SPB09]. We seek to provide better support for labeling full diagrams through the use of an automated system.

3. Our Approach

Our auto labeling technique is implemented as part of RATA (Recognition Algorithm Tools for ink Applications) [CBP12], a recognizer generation toolkit. It contains components to collect digital ink, manually label the ink data, select and generate stroke feature sets and a data mining tool wrapper that trains and generates recognizers. The manual labeling component and data mining features provide a suitable basis to implement the auto labeling, as our technique requires some manual labeling at the beginning and also training to produce a recognizer. The integration of auto labeling as a feature in the manual labeling interface also allows us to easily compare the two methods.

The auto labeling process begins with a collection of unlabeled digital ink sketches that can be drawn by various contributors. Our labeling technique can be used on isolated components (e.g. individual gestures) or full diagrams (e.g. entity relationship diagrams). Each component must be drawn as a single stroke, for example a rectangle must be drawn in one stroke rather than four. A few strokes from each class are manually labeled first. These strokes are used to train a recognizer using the built in RATA model generation component.

The recognizer is built using the default feature set and RATA.Gesture algorithm [CBP12]. Rata.Gesture is an ensemble of four algorithms: Bayesian Network [Ben07], Logit Boost [FHT00], Logistic Model Tree [LHF05; SFH05], and Random Forest [Bre01]. RATA.Gesture was developed through a rigorous analysis of nine data mining algorithms, which involved parameter tuning, feature

selection and the testing of ensembles. An evaluation shows that it outperforms other recognizers [CBP12]. The same algorithm can be used via the RATA model generation component to generate a robust recognizer with a fully labelled dataset.

The generated recognizer is then used to classify the remainder of the dataset. The training of the interim recognizer and labeling of strokes by the recognizer is automated; the user is only required to manually label a small subset of the data and then click a button to do the rest. Given that recognizers very seldom produce 100% accurate results, once the auto labeling is finished, the user checks for any labeling errors and corrects them manually. The auto labeling process is summarized in Figure 2.

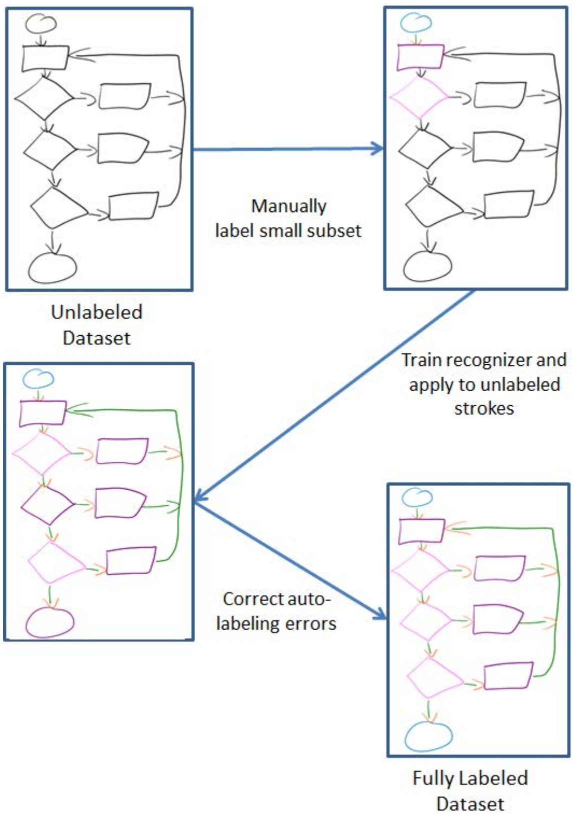

Figure 2: *The auto labeling process.*

3.1 Auto Labeling Accuracy

A central question to this automated process is how much of the original dataset must be manually labeled before the accuracy of the auto labeler is high enough for it to be useful. We investigated the accuracy of our auto labeling technique over different numbers of manually labeled examples using three different datasets (Figure 3). These datasets represent realistic sketch data that is difficult to manually label. The datasets have an average of 634 strokes, 5 classes and 20 contributing participants each, and two were also fully drawn diagrams with intersecting strokes and text.

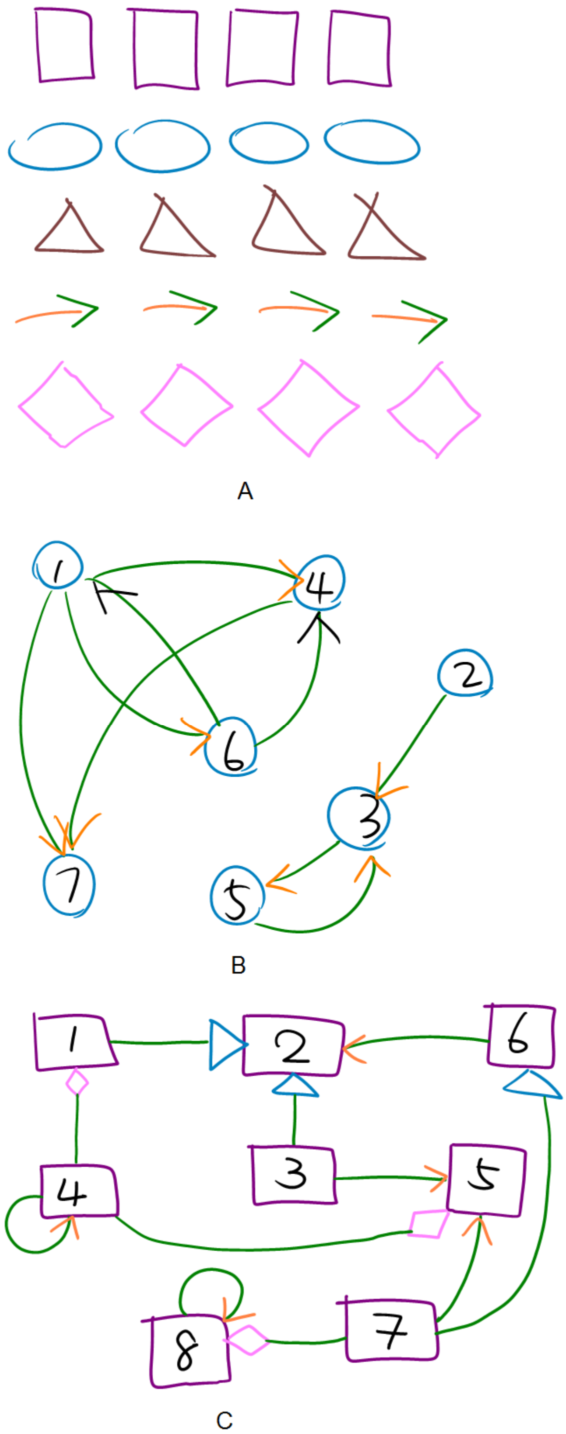

Figure 3: *Datasets used in auto label accuracy tests. A: Shapes, B: Directed Graph, C: Class Diagram*

The datasets were first fully manually labeled and checked to be correct in order to provide ground truth data for measuring accuracy. Next, unlabeled copies of each dataset were used to run automated auto labeling tests. The independent variable was the number of manually labeled

examples for each class and the dependent variable was the overall accuracy of the dataset after auto labeling. We set the independent variable to begin at 1 (i.e. one example of each class is manually labeled) and incremented it by 1, up to 22 (i.e. 22 examples of each class are manually labeled). At each increment, the accuracy of the auto labeler was recorded (Figure 4).

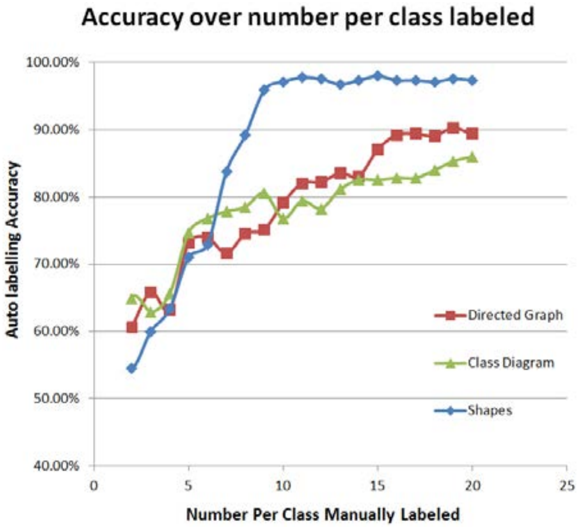

Figure 4: *Accuracy of auto labeler with increasing number of examples manually labeled per class*

The accuracy plots of the Directed Graph, Class Diagram, and Shapes datasets roughly follow logarithmic trend lines (coefficient of determination R^2 = 0.95, 0.91. and 0.87 respectively) suggesting diminishing returns of auto labeling accuracy with increasing numbers of manually labeled examples. From our observation of these curves we estimate that manually labeling 8 to 10 examples per class to be optimum; after that the slope of the curve levels off, reducing the accuracy gain. This is consistent with experiments on other trainable recognizers [Rub91, FPJ02].

With our experimental datasets this is 10-12% of the dataset but the number of labeled examples required is independent of the dataset size. For datasets with a similar number of examples per class (i.e. our shapes dataset), this will provide near 95% accuracy in auto labeling. This means that having manually labeled 10% of the dataset, 90% of the data will be automatically labeled with an accuracy of approximately 95%, therefore requiring only minor corrections to be made.

However, there are factors that affect the accuracy of the auto labeling and the number of manually labeled examples required. These factors include: the number of classes, the size of the dataset, the number of contributors to the dataset (who may have different drawing styles), the nature of the data itself, and how distinct each class is.

We investigated whether picking examples to manually label from a variety of different contributors would increase the auto labeling accuracy. Intuitively, we expected that providing labeled examples from as many different contributors as possible would produce a more style tolerant recognizer (i.e. if manual labeling ten instances of each class, we label one instance of each class from ten different contributors). We tested this hypothesis using the same method as above, but taking only one labeled example of each class per participant. We found that the accuracy only improved slightly and curved off into similar diminishing returns. This style of manual labeling incurs the cost of having to change class and/or participants for each stroke. This is much more time consuming than systematically labeling all instances of each class in turn on a single diagram before moving onto the next diagram.

3.2 Improving Manual Labeling

We also made some improvements to the existing manual labeling component in RATA; firstly, to reduce the time and effort required to manually label examples for auto labeling; secondly, to improve basic manual labeling as much as possible for a fairer comparison against auto labeling.

We observed that users tend to label all strokes of a single type (e.g. all circles and then all squares) so we modified the label selection to a mode selection: once a label is selected, any strokes selected while in this mode are given that label, until the next label is selected and the mode is changed (Figure 5).

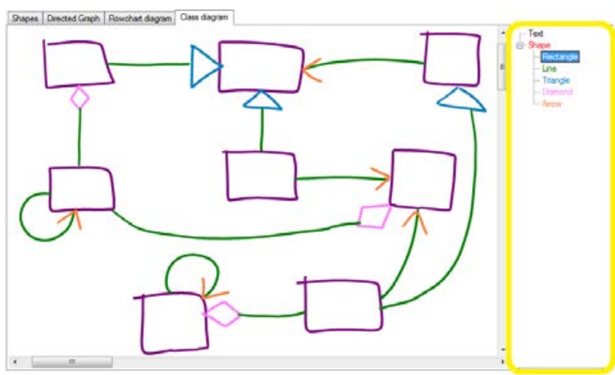

Figure 5: *Labeling component with labels area highlighted, currently set to 'Rectangle' mode.*

Stroke selection was originally accomplished by using the default Microsoft.Ink selection tools (i.e. point and click or a selection lasso). The basic point and click method is very difficult because strokes are too thin. Lasso selection causes difficulty when strokes contained other strokes, and thus unwanted strokes are also selected when lassoing the perimeter (Figure 6 A). We implemented an additional way to select strokes that allows users to draw selection strokes - any strokes that the selection stroke intersects with are selected (Figure 6 B).

Our informal tests showed that these changes made considerable improvements to the interaction required by manual labeling. The modal label selection style reduced the number of clicks required by users and shortened the

distance traveled by the mouse cursor as each label from the right hand side panel was only selected once for each diagram. The changes to stroke selection speeded up the labeling process and allowed for more accurate stroke selection.

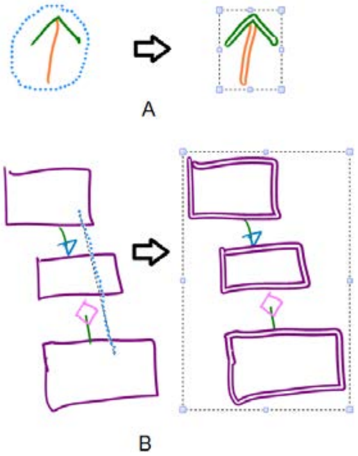

Figure 6: *A - The old lasso selection. B - The new intersecting stroke selection.*

4. Comparative Evaluation

In order to gauge the efficacy of our auto labeling technique, we compared the function to manual labeling in RATA [CBP12]. To do this, we ran a user experiment involving the two labeling methods to compare:

- Accuracy of labeling and error correction
- Time taken to label
- User preference

We recruited postgraduate computer science and software engineering students for the study because of their likelihood to understand sketch recognition and labeling tasks. Two subjects participated in pilot studies and nine subjects were recruited for the final experiment. Each experiment was performed on a desktop PC with a Core 2 Duo e8400, 3 GHz processor, 4GB RAM running 64bit Windows 7 and Morae screen capture software.

4.1 Tasks

The experiment involved the manual and auto labeling of data in the RATA labeling component. In order to compare the two labeling methods, we prepared two typical datasets for labeling: a flow chart diagram and a class diagram set (Figure 7). The datasets were completely unlabeled. Each dataset consists of 20 diagrams each drawn by a different person. In total they have a similar number of strokes and classes (Flow Chart: 599 strokes and Class diagram: 596 strokes. Both have five label classes; e.g. Rectangle, Triangle, Diamond, Line and Arrow for Class diagrams).

Each participant performed two labeling tasks: manually labeling one whole dataset, and auto labeling the other. In order to reduce the learning effect we rotated the order of datasets and labeling methods for each participant in the experiment, such that there were four unique conditions. Two participants did each condition, except for the last condition, which was completed by three participants. For manual labeling, users were asked to manually label every stroke of every diagram. For auto labeling, users were asked to manually label the first three diagrams (approximately 8 examples per class), activate the auto labeling function to auto label the other diagrams, and then review the diagrams to correct any labeling errors.

After each task was complete, participants were asked to fill in a questionnaire about the labeling method used. Each question was presented on a 5-point Likert scale. We asked participants to rate the labeling methods based on factors such as ease of use, intuitiveness, and utility. After a participant completed both labeling tasks, we also asked for general feedback on the labeling techniques, including which labeling method they found the easiest and which was most preferred.

During the pilot studies it was evident how inherently boring labeling is, with participants taking little notice of whether their labels were correct. In the study, to motivate participants, we offered a $20 prize for the 'best' labeling, best being fastest with the least errors (each error added 20 seconds to their time).

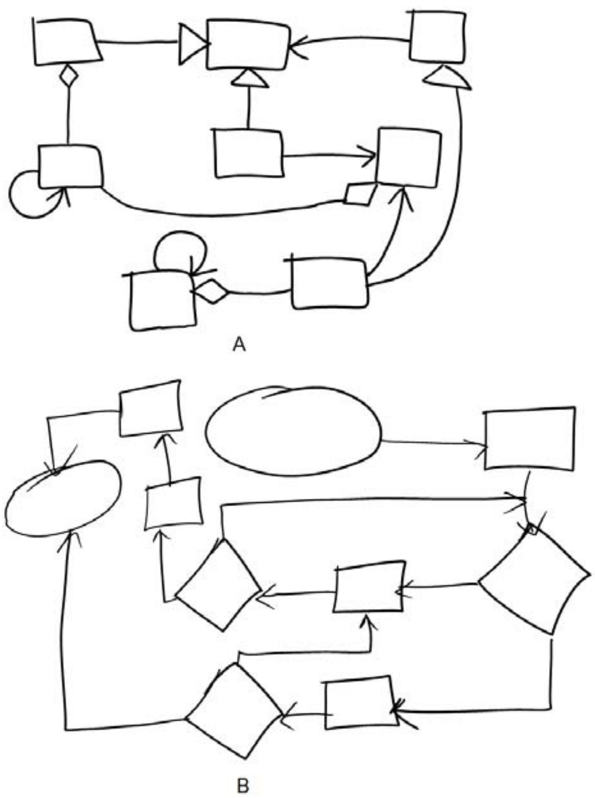

Figure 7: *Unlabeled datasets used in the comparative evaluation. A: Class Diagram, B: Flow Chart.*

4.2 Experimental Measures

Both performance and accuracy are important factors when labeling. The two central goals for improving labeling methods are to reduce the time taken to label and increase the accuracy of labeling. To evaluate these factors for our labeling schemes we captured the following data.

First, we recorded the time taken to label the datasets using each labeling condition. For manual labeling we measured the start time from when the user selects the first label to begin labeling the first diagram, and recorded the end time from when the last unlabeled stroke of the last diagram was labeled. For auto labeling, we recorded the start time in the same manner as manual labeling, the start time for each subsequent phase (recognizer generation, auto labeling and corrections) and took the end time from when the user affirms that they had finished correcting all the auto labeling errors.

The second measure we took was the accuracy of each labeling method. We measured this by counting the number of errors made by each labeling technique. In manual labeling, the two types of errors made are incorrectly assigned labels, and strokes that have not been labeled. In auto labeling, all strokes have labels as the auto labeler applies labels to everything. There are, however, errors made by the auto labeler that have not been checked and corrected by the user. We hypothesized that more errors would result from uncorrected misclassifications made by the auto labeler than from manual labeling errors, as checking errors made by the auto labeler maybe less intuitive.

4.3 Results

The first two pilot experiments revealed errors in the experimental and questionnaire set up. These were corrected and used for the final nine participants. All labeling errors and time data was recorded via Morae screen capture and analyzed after the experiment. Each experiment took 45 minutes on average to complete.

4.3.1 Preferred labeling technique

All nine participants rated auto labeling as the easier labeling method over manual labeling. Seven out of nine participants preferred auto labeling over manual labeling, while the other two recorded no preference.

Participants strongly agreed that they had a high understanding of both tasks (Mean=4.89, SD=0.33 for both tasks). They also enjoyed auto labeling more (M = 3.00 for manual labeling and M = 3.44 for auto labeling). Participants found auto labeling to be more helpful for completing the task, easier and more intuitive than manual labeling. (M = 4.44 for auto labeling for both questions versus M = 3.00 and M = 3.44 for manual labeling). Confirming our intuition, users found it easier to correct manual labeling errors than auto labeling errors (Manual: M = 4.33, SD = 0.5, Auto: M =3.56, SD = 0.7). Table 1 summarizes these statistics.

	Auto		Manual	
	Mean	**SD**	**Mean**	**SD**
Task Understanding	4.89	0.33	4.89	0.33
Task Enjoyment	3.44	1.13	3.00	1.22
Helped Complete Task	4.44	1.01	3.00	1.58
Easy and Intuitive	4.44	0.53	3.44	1.24
Easy Error Correction	3.56	0.73	4.33	0.50

Table 1: *Summary of user preference data.*

4.3.2 Performance and Accuracy

On every experimental trial, participants labeled faster with auto labeling than manual labeling. Participants on average took 539.6 seconds to auto label and 1003.7 seconds to manually label. With the average dataset at 597.5 strokes, that is 1.1 strokes per second for auto labeling and 0.60 strokes per second for manual labeling. A two-tailed paired-sample t-test of the mean time taken for each labeling strategy shows that auto labeling is significantly faster than manual labeling ($t = 5.206$, $p = 0.001$, $df = 8$). Table 2 summarizes these statistics.

	Error Rates		Time (seconds)	
Participants	**Auto**	**Manual**	**Auto**	**Manual**
1	1	6	632	868
2	1	2	464	918
3	11	0	425	911
4	5	9	511	754
5	3	3	536	1382
6	4	6	537	1054
7	3	7	672	995
8	1	4	559	713
9	3	15	520	1438
Mean	3.56	5.78	539.56	1003.7
SD	3.13	4.41	76.25	253.86

Table 2: *Summary of the evaluation performance data.*

The performance time for auto labeling was further broken down into its procedural parts (Table 3) and analyzed. We consider the manual labeling and recognizer generation time a constant factor in auto labeling and the recognizer labeling and user error correction time as a variable cost depending on the number of strokes in the dataset. We can use this information to estimate the performance times of manual labeling and auto labeling in different sized datasets. For manual labeling, given that our participants could label 0.60 strokes per second on average, we estimate that it takes 1.67 seconds (1 / 0.60) per stroke to label. The total time to manually label and to auto label a

dataset is calculated as follows (where t is the time and n is the number of strokes in the dataset):

$$\text{Total } t_{\text{manual labeling}} = 1.67n$$

$$\begin{aligned}\text{Total } t_{\text{auto labeling}} &= t_{\text{manual labeling}} + t_{\text{recognizer generation}} + (t_{\text{recognizer labeling}} + t_{\text{error correction}})n \\ &= 188 + 11 + (0.10 + 0.44)n \\ &= 199 + 0.54n\end{aligned}$$

With these two time estimates, we plot the estimated time costs for different sized datasets for both manual and auto labeling (Figure 8). The graph shows that the time to manually label strokes increases at a faster rate than the time to auto label strokes as the size of the dataset increases.

	Mean (sec)	SD
Manual Labeling Time	188	33.44
Recognizer Generation Time	11	2.5
Recognizer Labeling Time	59 (0 .10 sec per stroke)	1.5
User Error Correction Time	262.56 (0.44 sec per stroke)	76.20

Table 3: *Auto labeling performance time (seconds) broken up into its procedural components.*

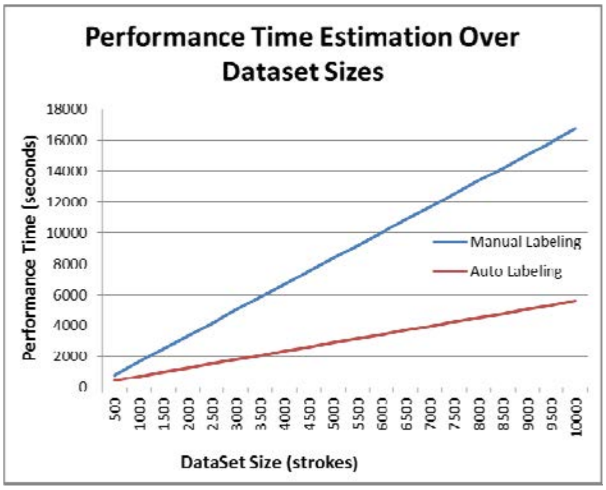

Figure 8: *Graph of estimated performance times over increasing dataset sizes.*

In terms of the number of errors found, the mean number of errors was 3.56 for auto labeling (after user corrections) and 5.78 for manual labeling. Seven participants had more errors in manual labeling than in auto labeling, one had the same amount of errors and one had more errors in auto labeling. The standard deviation was high however (SD = 3.13 for auto label and SD = 4.41 for manual label), and a 2-tailed Wilcoxon Signed-Rank test showed no statistically significant difference between the two mean error rates ($Z = -1.542$, $p = 0.123$). Table 2 summarizes these statistics.

4.3.3 General Feedback

We asked participants to contribute general feedback and suggestions on the labeling techniques. Most users liked the auto labeling technique because of the time and effort saved. One participant commented that auto labeling "saves time and energy from having to repetitively label each stroke", another mentioned "it was faster and only required a bit of my time". Other participants commented on the good accuracy of the auto labeler. An area identified as needing improvement is the speed of the auto labeling process in terms of the time taken for the algorithm to train and auto label the remaining diagrams. Some participants also commented about the error correction: "It's a bit harder to spot the mistakes"; "it was harder to check for errors because it was already all labeled".

While all participants preferred auto labeling to manual labeling, one participant commented that if there were less than five diagrams, they would prefer manual labeling because they find "checking the mistakes that the auto labeling may have made it a bit harder". However, this participant also commented that "if there are a lot to label, auto labeling would obviously help".

5. Discussion

The results of the experiments showed the effectiveness of an automated method of labeling data where it performed almost twice as fast as manual labeling. In addition auto labeling was the easiest, most intuitive and preferred method by our participants. The main source of errors for manual labeling was from strokes that had not been labeled. Due to the size and nature of full diagrams, it is not uncommon to mis-label certain stroke components. Since our auto labeling method attempts to label every stroke, it is not prone to this type of error.

A function which identifies unlabeled strokes for the manual labeler would alleviate this problem. If we exclude the errors resulting from unlabeled strokes, we expect the errors from auto labeling to be the same or higher than manual labeling. This is reflected by user complaints of the difficulty of checking for mistakes in data that is already labeled, and in the lower rating on "ease of error correcting" than manual labeling.

A possible method for identifying mis-labeled strokes from the auto labeler would be to re-build the recognizer with the fully labeled, manually corrected dataset. Then use the new recognizer to identify the data. Any cases where the recognizer and label differ could be highlighted for manual inspection. This theoretically should highlight some of the mis-labeled strokes; a number of experiments would be required to test this hypothesis.

High error rates from the auto labeler are an indication that the gesture set is difficult to distinguish. If researchers can isolate problematic gesture classes early on in the recognizer development process then we believe they are more likely to devise solutions to such problems when designing their algorithms.

The performance time is significantly lower for auto labeling, and we have estimated that this difference increases as the amount of data increases (Figure 8). These estimates do not factor in extra time taken due to fatigue or boredom when labeling or correcting large datasets. However, we expect these factors to influence the time taken to manually label a dataset more than auto labeling – where only the correction time might increase. In smaller datasets however, manual labeling is still the preferred method of labeling because there is a near constant cost of the actual auto labeling processing time. This was reflected in user comments as mentioned previously: one participant commented they would prefer manual labeling if there were less than five diagrams. In general, auto labeling is the most preferred and more beneficial method of labeling when dealing with moderate to large sized datasets (i.e. several hundred strokes and greater).

The results of our evaluation showed that participants did not enjoy auto labeling much more than manual labeling. One factor that may have affected this was that users are idle while waiting for the recognizer to generate and the auto labels to be applied. In our experiments this process took 70 seconds on average. This waiting time could be reduced by generating the recognizer and then applying labels in a separate thread. This would allow users to begin checking and correcting labels as soon each diagram has been labeled – rather than waiting for the whole process to end.

Our current auto labeling algorithm is designed for recognition of single stroke gestures. Due to this limitation our evaluation was performed on data without text. A potential way to extend our auto labeling scheme to include text is to incorporate a text shape divider into the auto labeling process. The process could incorporate either a hardcoded text shape divider algorithm or a domain-less trained divider, to first divide all text and shapes. Next, the normal auto training process is run on the full dataset with text included, and the results of the two different labeling schemes can be corroborated (i.e. agreement and disagreement from both auto labeling methods affect the outcome of the final label classification). Furthermore, instead of a single classification outcome, the recognizers can be tweaked to give a ranked classification and allow the user to have the final say in the labeling. This would also reduce the difficulty in checking for auto labeling errors as the system can identify strokes where there is low confidence in the classifications made.

Another limitation of our auto labeling technique is that it only deals with single stroke primitives. Work is currently in progress that attempts to apply the same data mining techniques in single stroke recognizers for grouping strokes together, allowing multi-stroke shapes to be recognized. For example, currently a rectangle must be drawn in one stroke, whereas a multi stroke recognizer would allow a rectangle to be drawn with four strokes. By the same principle, we can apply this technique for auto labeling and auto grouping of multi stroke shapes and glyphs.

Despite these limitations our auto labeling technique has been successful at reducing the time and effort required for labeling sketch datasets.

6. Conclusion

We designed and implemented an auto labeling method for single stroke primitives. Improvements have also been made to manual labeling through better label and stroke selection techniques. We compared auto and manual labeling and found that auto labeling performs faster, requires less effort, and users find it easier, more intuitive and preferable.

7. Acknowledgements

This research is supported by a Royal Society of New Zealand Marsden Grant and Rutherford Foundation Post-Doctoral Fellowship.

References

[AFGP08] AVOLA D., FERRI F., GRIFONI P., PAOLOZZI S.: A Framework for Designing and Recognizing Sketch-Based Libraries for Pervasive Systems. UNISCON (2008), Springer, 405-416.

[Ben07] BEN-GAL I.: Bayesian Networks. Encyclopedia of Statistics in Quality and Reliability, F. Ruggeri, F. Faltin and R. Kenett, John Wiley & Sons, 2007.

[BLMM07] BICKERSTAFFE A., LANE A., MEYER B., MARRIOTT K.: Developing Domain-Specific Gesture Recognizers for Smart Diagram Environments. GREC-IAPR Workshop on Graphics Recognition (2007), Curitiba, Brazil, 145-156.

[BPGW08] BLAGOJEVIC R., PLIMMER B., GRUNDY J. WANG Y.: A Data Collection Tool for Sketched Diagrams. Sketch Based Interfaces and Modeling (2008), Annecy, France, Eurographics, 73-80.

[Bre01] BREIMAN L.: Random Forests. Machine Learning 45(1), (2001), 5-32.

[BSP09] BLAGOJEVIC R., SCHMIEDER P., PLIMMER B.: Towards a Toolkit for the Development and Evaluation of Sketch Recognition Techniques. Intelligent User Interfaces Sketch Recognition Workshop (2009), Florida, USA.

[CBP12] CHANG S. H.-H., BLAGOJEVIC R., PLIMMER B.: RATA.Gesture: A Gesture Recognizer Developed using Data Mining. Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM), 26(3), (2012), accepted, in press.

[FGP*09] FIELD M., GORDON S., PETERSON E., ROBINSON R., STAHOVICH T., ALVARADO C.: The effect of task on classification accuracy: Using gesture

recognition techniques in free-sketch recognition. CAD/GRAPHICS, 34(5), (2009), 499-512.

[FHT00] FRIEDMAN J., HASTIE T., TIBSHIRANI R.: Additive Logistic Regression: a Statistical View of Boosting. The Annals of Statistics, 28(2), (2000), 337-407.

[FPJ02] FONSECA M. J., PIMENTEL C. E., JORGE J. A.: CALI: An Online Scribble Recogniser for Calligraphic Interfaces. AAAI Spring Symposium on Sketch Understanding, (2002), IEEE, 51-58.

[Joh09] JOHNSON G.: Picturephone: A game for sketch data capture. Intelligent User Interfaces Workshop on Sketch Recognition, (2009), Sanibel Island, Florida.

[JD09] JOHNSON G., DO E. Y.-L.: Games for sketch data collection. Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling (2009), New Orleans, Louisiana, ACM, 117-123.

[KJM*09] KASTER B., JACOBSON E., MOREIRA W., PAULSON B., HAMMOND T.: SOUSA v2.0: Automatically Generating Secure and Searchable Data Collection Studies. International Workshop on Visual Languages and Computing, (2009), Redwood City, CA, USA, 365-368.

[LHF05] LANDWEHR N., HALL M., FRANK E.: Logistic Model Trees. Machine Learning, 59(1-2), (2005), 161-205.

[MTL*09] MACLEAN S., TAUSKY D., LABAHN G., LANK E., MARZOUK M.: Tools for the efficient generation of hand-drawn corpora based on context-free grammars. Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling (2009), New Orleans, Louisiana, ACM, 125-132.

[MMBK09] MEYER B., MARRIOTT K., BICKERSTAFFE A., KNIPPING L.: Intelligent diagramming in the electronic online classroom. Human System Interaction (2009), Catania, 174-180.

[PWJH08] PAULSON B., WOLIN A., JOHNSTON J., HAMMOND T.: SOUSA: Sketch-based Online User Study Applet. Sketch Based Interfaces and Modeling, (2008), Annecy, France, Eurographics, 81-88.

[Rub91] RUBINE, D. H.: Specifying gestures by example. Proceedings of SIGGRAPH (1991), ACM, 329-337.

[SPB09] SCHMIEDER P., PLIMMER B., BLAGOJEVIC R.: Automatic Evaluation of Sketch Recognizers. Sketch Based Interfaces and Modelling, (2009), New Orleans, USA, 85-92.

[SKN07] SIGNER B., KURMANN U., NORRIE M. C.: iGesture: A General Gesture Recognition Framework. 9th International Conference on Document Analysis and Recognition (2007), Curitiba, Brazil, 954-958.

[SNK07] SIGNER B., NORRIE M. C., KURMANN U.: iGesture: A Java Framework for the Development and Deployment of Stroke-Based Online Gesture Recognition Algorithms. Zurich, Switzerland, Institute for Information Systems, ETH Zurich (2007).

[WSA07] WOLIN A., SMITH D., ALVARADO C.: A Pen-based Tool for Efficient Labelling of 2D Sketches. 4th Eurographics Workshop on Sketch-Based Interfaces and Modeling, (2007), Riverside, CA, 67-74.