Supervised Machine Learning for Grouping Sketch Diagram Strokes

Philip C Stevens*, Rachel Blagojevic+, Beryl Plimmer†

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

Abstract

Grouping of strokes into semantically meaningful diagram elements is a difficult problem. Yet such grouping is needed if truly natural sketching is to be supported in intelligent sketch tools. Using a machine learning approach, we propose a number of new paired-stroke features for grouping and evaluate the suitability of a range of algorithms. Our evaluation shows the new features and algorithms produce promising results that are statistically better than the existing machine learning grouper.

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.

Keywords: Digital Ink recognition, grouping strokes

1. Introduction

Grouping strokes in a hand-drawn sketch into semantically meaningful entities is a difficult problem for sketch recognition. Often there are no obvious spatial or temporal boundaries between the entities within a sketch to assist in solving this problem. Some methods [Rubine 1991; Gennari et al. 2005] place constraints on the way a user must sketch to ensure the boundaries are clear and distinct. Our goal is to support natural sketching without placing such constraints on the user. To do this a more accurate way of grouping strokes is needed. We extend the feature-based grouping strategy proposed by Peterson et al [2010] as a possible strategy for better grouping.

Peterson et al's [2010] strategy utilizes two classifiers on the diagram: a divider and a grouper. The divider classifies the strokes according to the type of object they are. The grouping classifier is then applied separately to the sets of strokes output from the divider to find which pairs of strokes belong to the same group. Clusters of strokes are then formed by adding a stroke to a cluster if the grouper pairs it with any one of the strokes already in the cluster.

This strategy is distinct from other grouping strategies as it reduces the problem to classification, whereas most other strategies [Shilman et al. 2004; Ouyang et al. 2009; Hammond et al. 2011] perform a complex search to find the groups.

We have chosen to explore this grouping strategy for two reasons. First, it is a promising approach in terms of effectiveness and scalability. Other approaches to stroke grouping are limited in the domains they can be applied to, or do not scale well to more complex or dense diagrams. Second, it is the least explored strategy for grouping with the only instance found described by Peterson et al [2010].

This machine learning approach critically depends on a set of features extracted from the sketch and supervised machine learning algorithm classification. We extend the exploration of feature-based grouping with the design of a number of new features, and a systematic analysis of selected algorithms to find which, if any, are the most promising for grouping.

The remainder of this paper is structured as follows: Section 2 provides a summary of previous work on sketch grouping techniques. Section 3 proceeds to describe the methodology employed in this research. Section 4 describes extensions made to RATA [Chang et al. 2012], a software platform used for developing sketch recognisers. Section 5 describes the feature set compiled, developed, and implemented. Section 6 describes the sketch data collected and used for the algorithm analysis. Section 7 describes the range of algorithms used for grouping classification, as well as the analysis performed on the collected sketch data. Section 8 shows the results of the algorithms and a statistical analysis to identify the top performers. Section 9 presents the implications of this work in the wider field of sketch recognition and grouping. Section 10 concludes the paper and suggests directions for future work.

2. Related Work

Many existing recognition systems avoid the challenge of grouping by placing constraints on the way a user sketches. One commonly employed method is to restrict the objects of recognition to single strokes [Rubine 1991; Wobbrock et al. 2007; Paulson et al. 2008; Chang et al. 2012]. Other systems require the user to delimit each sketch object by providing a temporal pause between them [Druin et al. 1999; Hse et al. 2004] or require each symbol to be a contiguous sequence of strokes [Gennari et al. 2005]. These approaches are relatively easy to implement and the constraints they impose aid recognition accuracy. However, such methods are not always practical, and do little to promote an environment of unconstrained and fluid sketching [Fonseca et al. 2002; Alvarado et al. 2007] a significant goal for many sketch tools.

Other grouping techniques do not impose on the way the user sketches, these can be described by four categories: context-based, perception-based, search-based, and feature-based. Many of these techniques simultaneously recognize and group sketch elements.

* phillipsteven4@gmail.com
+ rachel.blagojevic@auckland.ac.nz
† beryl@cs.auckland.ac.nz

SBIM 2013, June 19 – 21, 2013, Anaheim, California.

Context-based techniques extract a set of context-specific markers from the sketch. Kara and Stahovich [2004] developed an approach to grouping in network diagrams that involves first identifying arrows using a few characteristic features such as pen speed, angles and length. Any ink not classified as an arrow is then clustered into possible components which are then identified using a trainable recognizer. One limitation of their system is that arrows must be drawn in a specific manner. Shilman et al [2003] use a similar approach in the domain of document analysis, dividing strokes as belonging to text or shapes. These approaches are context specific and depend on reliable markers in the domain.

Perceptual-based techniques utilize a set of laws of perceptual organization devised by Gestalt psychologists. These laws explain how humans derive order from visual scenes and are based on the concept that people perceive using "invisible extensions as genuine parts of the visible" [Arnheim 1997]. Examples of these laws include proximity, similarity, symmetry, continuation, and closure of elements involved [Kanizsa 1979].

Saund et al [2003] make use of these Gestalt principles to first decompose a sketch into text and contiguous line segments. The principals used rely on the idea that if one imagines the sketch viewed from a distance then the text will appear as "blobs" and the contiguous line segments will appear as single continuous lines. Gestalt principles are applied again to group objects into even larger structures. This approach is designed more for interactive manipulation of sketches rather than strict object recognition. Other examples that use similar Gestalt principles to cluster strokes include PerSketch [Saund; et al. 2002] and ScanScribe [Saund et al. 1994].

Search-Based methods, work by searching over the set of possible candidates of stroke groups. Shilman and Viola [2004], for instance, utilize A-star search over the set of stroke groups. They then take the rendered image of the group of strokes and a set of stroke features such as curvature, and use this information to distinguish the valid groupings from meaningless clusters of strokes. Various heuristics can be used to guide search such as probabilities produced by Baysian networks [Alvarado et al. 2007]. Ouyang and Davis [2009] and Hammond and Paulson [2011] also make use of heuristic guided search.

Feature-based methods compute a set of quantifiable attributes from the strokes and use these features to classify and group the strokes. The features may include properties such as length, angle, and aspect ratio, as well as temporal and spatial context. These approaches train a classifier to perform the recognition on the candidate feature vectors [Rubine 1991; Blagojevic et al. 2011; Chang et al. 2012].

The only feature-based method to grouping found is described by Peterson et al [2010]. They propose a two-step process: first strokes are divided into nodes, edges, and text. Then each class of strokes is grouped. They try two methods for grouping: a simple threshold approach and a more sophisticated grouping classifier. The simple approach uses a distance threshold to determine if strokes belong to the same group. The second approach uses the AdaBoost algorithm (with decision trees) trained with 13 features – mostly measuring spatial context. They evaluated their grouping classifier on digital circuits and family trees. The AdaBoost classifier is slightly more accurate than their simple threshold based approach. They report an accuracy of between 88% and 93% of perfectly clustered strokes when the grouping classifiers are provided with the correct stroke classes.

Feature based approaches have been successfully used in other areas of sketch recognition. Blagojevic et al [2011] performed a systematic analysis of both the features and the classifiers used in the domain of text-shape stroke division. The features were collected across the related literature, and a number of new features were implemented. Seven classifiers were analysed with various algorithm tunings and ensembles included. The results of this analysis were text-shape dividers with a higher rate of accuracy than any existing ones. Chang et al [2012] applied a similar methodology to the domain of single stroke shape recognition, once again reporting higher rates of accuracy than existing recognizers. These demonstrate a useful methodology for developing better sketch recognizers.

This research extends the exploration of this technique by trialling other classifiers to be used for grouping, and by proposing a number of new paired-stroke features.

3. Our Approach

As an exemplar domain we perform grouping on graph-based diagrams. The strokes are pre-labelled as nodes or edges: we assume the output from a divider such as Blagojevic et al [2011] or Peterson et al [2010] that labels the strokes as belonging to either a node or edge.

We use the methodology of Blagojevic et al [2011]. This means using a large feature set and performing a systematic analysis of machine learning algorithms to determine which yield the best results. This methodology has proved to be very productive in developing not only superior feature-based recognizers, but better recognizers in general. It is believed that a similar methodology will improve feature-based grouping. In this case the learning algorithms are selected based on their performance in dividing and grouping classification. RATA [Chang et al. 2012] is used to aid with this process as it was designed with this methodology in mind, albeit for only single stroke recognition.

To apply this methodology several tasks are carried out.

- First, various functions are added to RATA [Chang et al. 2012] to permit the training and testing of grouping classifiers.
- Second, three new sets of diagrams are collected. And these together with two existing datasets are labelled.
- Third, the paired-stroke feature set described by Peterson et al [2010] is implemented and new paired-stroke features are formulated and implemented.
- Fourth, selected classification algorithms are evaluated using k-fold cross validation. The feature sets used by the algorithms are Peterson et al's [2010] and our extended feature set. The evaluations are run across the five labelled data sets.
- Finally the results are analysed to find which algorithms performed best at grouping.

4. Extensions to RATA

RATA is a software tool that supports digital ink recognition investigation and the generation of models for various algorithms [Chang et al. 2012]. The previous version of RATA included functionality for: data collection in the form of sketches, labelling of individual ink stroke data, a library of single stroke features, feature data set generation, an interface to Weka [Hall et al. 2009]

for model generation; an evaluation module for k-fold comparative evaluations, and a test interface for informal trials of recognizers generated by RATA.

This project required three extensions: labelling of stroke pairs as belonging to the same or different groups, generation of paired stroke features, and a grouper evaluator.

A new labeller has been added to RATA to label strokes as groups. Each group of strokes is labelled as a "node" or "edge" (these labels are flexible). From this labelling each pair of strokes can be labelled as belonging to the 'same group' or 'different group'. For example consider the strokes in Figure 1, the pairs ab, ac, and bc are labelled same group, while the pairs ad, bd, and cd are labelled 'different group'. Strokes are also labelled as being of class node or edge.

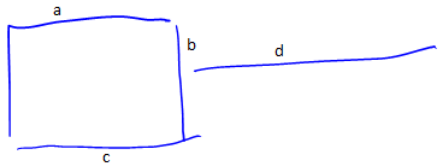


Figure 1 *The strokes a,b,c belong to the same group and are of class 'node', stroke d is a different group and is of class 'edge'*

The new feature generation produces a feature vector for each pair of strokes of the same class (i.e. ab, ac,bc but not ad, bd, cd as they are of different classes) as opposed to the individual strokes. These paired-stroke feature data sets are used to train the algorithms.

The grouper evaluator extends the manner in which RATA interfaces to Weka [Hall et al. 2009] for k-fold validation. Typically k-fold generation of single stroke data can split the strokes in any way. However with grouping, all the strokes from a single diagram must be in the same fold or correct recognition of a group may not be possible. Modifications were made to RATA's Weka interface to achieve this.

5. Features

Grouper features have a scant presence in related work. This is due to Peterson et al [2010] being the first to utilize feature-based techniques for grouping. Thus the 13 features they implemented form the foundation for the grouper feature set used in this research. Examining the features used by Peterson et al [2010] shows several seemingly significant characteristics of paired-strokes that are missing. During this process 11 new paired-stroke features were developed, 5 of them are modifications to Peterson et al's [2010] features while the other 7 are novel. In total 24 features are used.

5.1. Modified Peterson Features

Where Peterson el al [2010] compute the following features in respect to a single stroke, we compute them as stroke pairs. We have modified their tolerance for end- and mid-point intersections to be a pairwise function. Our tolerance function computed for each pair based on their lengths.

Let $L_{THR}(x,y)$ be the function that determines the threshold where x and y are the strokes, L_x and L_y are the arc lengths of the first and second stroke respectively and L_{SKETCH} be the average arc length of all the strokes in the sketch:

$$L_{THR}(x,y) = MIN\left(L_{SKETCH}, \frac{L_x + L_{SKETCH}}{2}, \frac{L_y + L_{SKETCH}}{2}\right) * 0.15$$

This makes L_{THR} determined by the shorter of the two strokes. Let L_{MIN} be the arc length of the shorter of the two strokes. If L_{MIN} is less than L_{SKETCH}, then L_{THR} will be $\frac{L_{MIN}+L_{SKETCH}}{2} * 0.15$. If L_{MIN} is greater than L_{SKETCH}, then L_{THR} will be $L_{SKETCH} * 0.15$. This means L_{THR} is made proportionately larger if the shorter stroke is smaller than the average stroke length and proportionately smaller if the shorter stroke is larger than the average stroke length. This prevents small strokes from receiving too small of a threshold, and large strokes from receiving too big of a threshold.

f1 We have modified their LL feature which measures the number of endpoint-to-endpoint intersection to use it in a paired manner. It measures the number of times the endpoints of a pair of strokes intersect or nearly intersect. As an example of the difference between our and Peterson's application of these features consider the strokes figure 1. Using Peterson's features stroke *a* would have two LL intersections (ab and bc). Whereas we computer features by pairs of strokes so the pair ab would have one and the pair ac would have one intersection.

f2 & f3 The next two features are modifications of their *XL-intersection* and the *LX-intersection* features that determine whether one or both of the endpoints on one stroke falls within L_{THR} of any non-endpoint on the other stroke. We do this as a pairwise computation.

f4&5 $Delay_{A,B}$ are likewise pairwise computation.

5.2. New Features

The next four features involve the concept of intersecting strokes.

f6 The first of these features measures the number of points at which the two strokes *intersect*. In Figure 2 the stroke pair has three intersection points

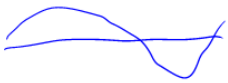


Figure 2 *Pair of strokes with three intersections*

f7 *Class Connection* (Figure 3) is a binary feature that determines if two strokes are connected through only node strokes or only edge strokes. This global relationship between the pair of strokes gives a higher level perspective of the intersection features.

Figure 3 *Stroke a and stroke b are Class Connected. Stroke a and stroke c are not Class Connected as there is an edge stroke dividing them.*

The bounding box of a stroke is very important to many of the features in single stroke feature sets. It may prove to be useful as well in determining key paired-stroke characteristics. For a pair of strokes two methods are conceived of for determining the bounding box area (Figure 4).

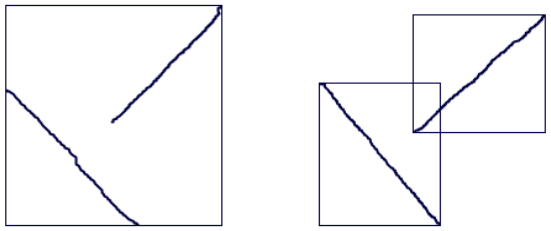


Figure 4: *Left demonstrates the combined bounding box area, whereas the right demonstrates the summed bounding box area.*

f8 First, one measures the area of the bounding box that contains both strokes. This is referred to as the *combined bounding box area*.

f9 Second, one is the sum of the areas of each strokes bounding box, minus any area of overlap between the bounding boxes. This is referred to as the *summed bounding box area*.

f10 The *bounding box ratio* is the summed bounding box area divided by the combined bounding box area. If the bounding box ratio equals one then the bounding boxes of individual strokes overlap exactly. As the bounding box ratio approaches zero the further apart the bounding boxes of the strokes are.

f11 Another useful feature may be the *density* of ink between the strokes. This is approximated by measuring the concentration of ink within the combined bounding box, and the ink in the two strokes themselves.

6. Data

Graph-based diagrams are used as an exemplar domain for evaluating our sketch groupers. They are selected as they represent an extremely challenging type of diagram to group due to the interspersed nature of the diagram and the close proximity of edges and nodes. In other words, the nodes and edges are not isolated spatially, and frequently overlap.

We have collected three new datasets and use two existing data sets that Peterson el al [2010] employed. Using the same datasets as Peterson el al [2010] allows us to better compare results.

Three types of graph-based diagrams were collected: flow charts (Figure 5), class diagrams (Figure 6), and digital circuits (Figure 7). These diagram types are chosen to ensure a large portion of multi-stroke data. Of note is that the flowcharts and class diagrams naturally include arrows, while the circuit diagrams do not.

Following the methodology of Chang et al [2012] participants are provided with instructions of what to sketch. To aid them a simple exemplar sketch is provided and a visual dictionary explaining the syntax for the particular diagram type. No drawing restrictions were placed on the participants: it is interesting to see different styles of arrow head within one diagram (Figure 6). The resulting diagrams were cleaned of extraneous strokes, and the strokes are labelled as nodes or edges and each group is identified to provide ground truth data.

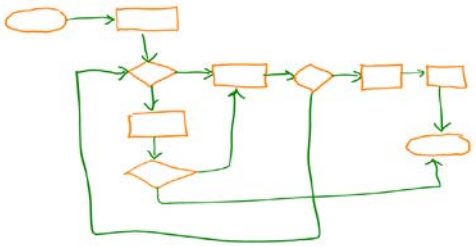
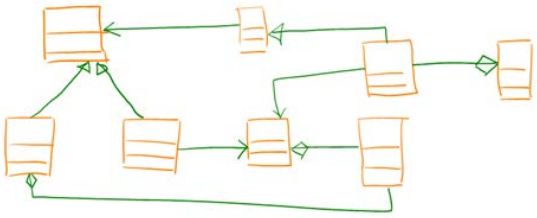


Figure 5 *Example of flow chart from dataset*

	Single Stroke Shapes	Multi-stroke Shapes	Shapes	Strokes
Node	117	93	210	370
Edge	34	215	249	512
Total	151	308	459	882

Table 1 *Summary statistics for flow chart dataset*

Figure 6 *Example of class diagram from dataset*



	Single Stroke Shapes	Multi-stroke Shapes	Shapes	Strokes
Node	0	174	174	472
Edge	11	179	190	750
Total	11	353	364	1222

Table 2 *Summary statistics for class diagram dataset*

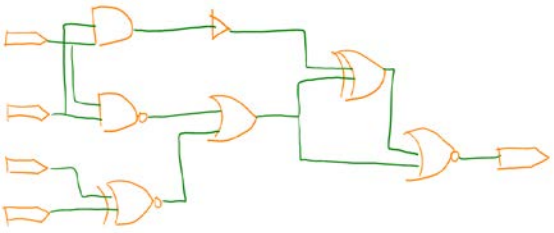


Figure 7 *Example of digital circuit from dataset*

	Single Stroke Shapes	Multi-stroke Shapes	Shapes	Strokes
Node	27	195	222	671
Edge	79	128	207	444
Total	106	323	429	1115

Table 3 *Summary statistics for digital circuit dataset*

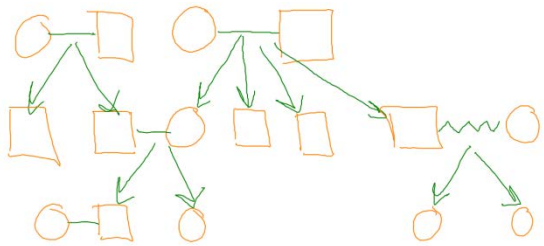


Figure 8 *Example of family tree from dataset*

	Single Stroke Shapes	Multi-stroke Shapes	Shapes	Strokes
Node	225	106	125	476
Edge	168	275	331	699
Total	393	381	456	1175

Table 4 *Summary statistics for family tree dataset*

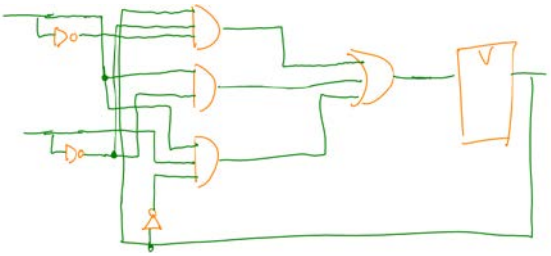


Figure 9 *Example of digital circuit from second dataset*

	Single Stroke Shapes	Multi-stroke Shapes	Shapes	Strokes
Node	1	476	477	1499
Edge	406	431	837	2267
Total	407	907	1314	3766

Table 5 *Summary statistics for second digital circuit dataset*

Twenty participants sketched each of the three diagrams. In total, the diagrams consist of 1513 strokes that form 606 nodes and 1706 strokes that form 646 edges. Of the nodes 462 are multi-stroke and of the edges 522 are multi-stroke. The breakdown for each type of diagram is provided in Table 1, Table 2, and Table 3 respectively.

In addition, two diagram sets that Peterson et al [2010] used are employed : a set of family trees and a set of digital circuits. The family tree data (Figure 8) is from the EtchaSketch[1] corpus, all drawn on a Tablet PC. As our goal is to test grouping, only diagrams consisting of more than 5 strokes were used yielding a total of 30 sketches from 29 users. The digital circuits (Figure 9) were collected by Alvarado et al [2007] and can be downloaded from their webpage[2]. To limit computation time only diagrams consisting of 200 strokes or less are used yielding 90 diagrams collected from 24 participants. The breakdowns for these diagrams are provided in Table 4 and Table 5.

7. Analysis

We tested 11 algorithms selected for the reasons described here. LogitBoost [Friedman et al. 1998], LadTree [Holmes et al. 2001] and AdaBoost (with C4.5 base classifier) were chosen because of their good performance in other digital ink classification problems [Peterson et al. 2010; Blagojevic et al. 2011]. We also selected other representative algorithms of different styles: C4.5 (Weka J48) and Decision Stumps are both Trees. Naïve Bayes is a Bayes classifier while K Nearest Neighbour (KNN – Weka lBK) is a lazy learner. Sequential Minimal Optimization (SMO) is a support vector machine. PART and Logistic are examples of linear classifiers.

Each of the algorithms was evaluated using the Weka [Hall et al. 2009] implementation with the default values. They were trained and tested as a grouper using the paired-stroke feature set described in section 5. For comparative results, AdaBoost is also trained using the feature sets used by Peterson et al [2010], we label this Peterson's. Doing this permits a comparison between Peterson's feature set and the extended feature set. Ten-fold cross validations are performed on whole diagrams – as described above all parts of diagrams must be in the same fold for grouping to be tested.

8. Results

The performances of the groupers are determined by the percentage of clusters that have no erroneous strokes. An erroneous stroke can be either an excess stroke or a missing stroke. These are computed on a per cluster basis, and then averaged across all shapes. The results are summarized in Table 6.

We ran chi-square tests to investigate if there were significant differences in the proportion of correct and incorrect groups of shapes over all datasets for different pairs of algorithms. There is a significant difference between Adaboost with the extended feature set and Peterson's (χ^2=100.67, df= 1, p<0.001). This indicates that our new features make a significant contribution to more accurate grouping.

There is also a significant difference between LogitBoost and Peterson's (χ^2=135.72, df= 1, p<0.001). This indicates that our combination of algorithm and extended feature set results in a significantly more accurate grouper than Peterson's.

Table 7 shows the results of chi-square tests on the difference in the proportion of correct and incorrect groups of shapes for the top six algorithm pairs. There is no significant difference between the performance of LogitBoost, LADTree, SMO, Logistic and AdaBoost. There is a significant difference between PART and the top three algorithms: LogitBoost, LADTree and SMO.

[1]http://rationale.csail.mit.edu/ETCHASketches/

[2]http://www.cs.hmc.edu/~alvarado/research/download.html

Data Set / Algorithm	Class	Digital Circuit(1)	Digital Circuit(2)	Family	Flow Chart	Overall %
LogitBoost	92.00	92.60	70.53	78.00	94.10	80.96
LADTree	90.20	93.50	69.73	78.20	92.90	80.37
SMO	92.70	90.90	71.18	73.50	90.60	79.87
Logistic	90.60	92.80	67.61	77.50	93.10	79.32
AdaBoost	93.90	93.50	66.21	77.30	93.60	79.25
PART	91.50	90.50	64.57	75.60	93.00	77.48
C4.5	92.00	91.10	61.54	76.10	89.30	75.82
KNN	92.70	88.80	53.75	70.80	91.60	71.74
Peterson	83.60	82.40	*	72.70	91.30	67.87
Decision Stump	87.00	90.90	26.62	49.30	87.40	55.67
NaiveBayes	58.10	14.30	16.91	46.30	44.80	30.17

Table 6 *Grouper Results for each dataset-algorithm, the percentage of correct clusters is shown with the overall percentage for each algorithm shown in the last column.*

(χ^2, p)	LogitBoost	LADTree	SMO	Logistic	AdaBoost	PART
LogitBoost	-	0.314, 0.576	1.089, 0.297	2.515, 0.113	2.720, 0.990	**11.117, 0.001**
LADTree	-	-	0.234, 0.629	1.053, 0.305	1.187, 0.276	**7.702, 0.006**
SMO	-	-	-	0.294, 0.587	0.367, 0.544	**5.255, 0.022**
Logistic	-	-	-	-	0.004, 0.949	3.063, 0.080
AdaBoost	-	-	-	-	-	2.845, 0.092

Table 7. *Chi-square tests results, df = 1, results in bold show significant differences between these algorithms*

9. Discussion

This research is significant as feature-based grouping methods are underexplored, and yet have potential as a general grouping strategy in terms of effectiveness and scalability. We have expanded the feature set proposed by Peterson et al [2010] to include 11 new features. Five of these are variations on their features that measure pair-wise intersections (rather than single stroke intersections) with a different threshold computed for each pair. The remaining six are novel features. We show that these new features are effective at increasing the recognition success rate with a significant difference between the Adaboost algorithm with the original (Peterson's) and expanded feature set.

Our results cannot be directly compared to those reported in Peterson et al [2010] for several reasons. We re-implemented their features to the best of our understanding. They undertake three-way dividing and grouping where as we excluded text, reasoning that there are already good text shape dividers and text groupers. Finally did not we use the entire datasets.

There is no significant difference between the top five algorithms we tested. Furthermore, note that the ranks of the algorithms differ between the datasets, these were not at statistically significant levels between the top five. PART ranked 4th for one data set while C4.5 ranked 4= and 5th for two other datasets. LADTree has consistently performed well for this type of problem. Blagojevic et al [2011] found it among the best performing algorithms for dividing writing and drawing and Chang et al [2012] included LADTree in the best performing algorithms. However Chang et al [2012] found an ensemble of four algorithms, Bayesian Network [Ben-Gal 2007], LogitBoost [Friedman et al. 1998], LadTree [Holmes et al. 2001] and Random Forest [Breiman 2001], outperformed any of the individual algorithms.

Given the varied performance of the tested algorithms across the datasets an ensemble may be worthy of exploration. The best ensemble may not be the top, say, five, algorithms, but may, as Chang et al [2012] found include the lower ranked algorithms that with their different strategies counterbalances the weakness of the other algorithms.

We have focused this project on feature-based machine-learning. This has the particular advantage of being domain independent. That is it could be applied to a new domain without any further coding required. To apply this method to a new domain one simply provides a set of examples for each type of diagram required. Furthermore we have concentrated on the grouping phases of the process, assuming divider results from a divider such as [Peterson et al. 2010; Blagojevic et al. 2011], both of which have very high recognition rates. In future work we intend to evaluate the performance of a full grouper that first applies a divider, and then applies the grouping classifier.

There are a variety of other grouping strategies, for example: search (e.g., Ouyang and Davis [2009]), perception (e.g., Saund, Fleet et al. [2003]) or contextual recognition (e.g., Kara and Stahovich [2004]). The optimal grouper for diagrams may be a hybrid of approaches. For example, perhaps a feature-based approach could be applied first to determine various candidate clusters, and then one of the other approaches could be used to fine-tune the clusters. Or a search technique could use the clusters as a starting point, and add or remove strokes to determine if anything more meaningful results.

Regardless of what the optimal approach may be, the development of more effective grouping strategies is important to

aid in the ability to recognize multi-stroke sketch elements. This in turn could support freely-drawn sketches in sketch tools such as SketchNode [Plimmer et al. 2010].

10. Conclusion

The primary objective of this research was to develop more accurate grouping classifiers. The results show that the features developed with the algorithms selected, outperform the earlier feature based grouper.

11. References

ALVARADO, C. AND R. DAVIS. 2007. Dynamically constructed Bayes nets for multi-domain sketch understanding. *ACM SIGGRAPH 2007 courses*, ACM. 33.

ALVARADO, C. AND M. LAZZARESCHI. 2007. Properties of Real World Digital Logic Diagrams. *1st International Workshop on Pen-based Learning Technologies*. 1-6.

ARNHEIM, R. 1997. Visual Thinking. Berkeley, University of California Press.

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

BLAGOJEVIC, R., B. PLIMMER, J. C. GRUNDY AND Y. WANG. 2011. Using data mining for digital ink recognition: Dividing text and shapes in sketched diagrams. *Computers & Graphics* **35**(5): 976-991.

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

CHANG, S. H.-H., R. BLAGOJEVIC AND B. PLIMMER. 2012. RATA.Gesture: A Gesture Recognizer Developed using Data Mining. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM)* **26**(3): 351-366.

DRUIN, A., J. MONTEMAYOR, J. HENDLER, B. MCALISTER, A. BOLTMAN, E. FITERMAN, A. PLAISANT, A. KRUSKAL, H. OLSEN, I. REVETT, T. P. SCHWENN, L. SUMIDA AND R. WAGNER. 1999. Designing PETS: a personal electronic teller of stories. *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, ACM. 326-329.

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

FRIEDMAN, J., T. HASTIE AND R. TIBSHIRANI. 1998. Additive Logistic Regression: a Statistical View of Boosting. *The Annals of Statistics* **28**(2): 337-407.

GENNARI, L., L. B. KARA, T. F. STAHOVICH AND K. SHIMADAA. 2005. Combining Geometry and Domain Knowledge to Interpret Hand-Drawn Diagrams. *Computers & Graphics* **29**(4): 547-562.

HALL, M., E. FRANK, G. HOLMES, B. PFAHRINGER, P. REUTEMANN AND I. H. WITTEN. 2009. The WEKA data mining software: An update. *SIGKDD Explorations* **11(1)**: 10-18.

HAMMOND, T. AND B. PAULSON. 2011. Recognizing sketched multistroke primitives. *ACM Trans. Interact. Intell. Syst.* **1**(1): 1-34.

HOLMES, G., B. PFAHRINGER, R. KIRKBY, E. FRANK AND M. HALL. 2001. Multiclass alternating decision trees. *ECML*: 161-172.

HSE, H. AND A. R. NEWTON. 2004. Recognition and Beautification of Multi-Stroke Symbols in Digital Ink. *AAAI Fall Symposium Series*, Oct. 2004. pp. 78-84.

KANIZSA, G. 1979. Organization in vision: essays on gestalt perception. *Praeger*.

KARA, L. B. AND T. F. STAHOVICH. 2004. Hierarchical Parsing and Recognition of HandSketched Diagrams. *UIST '04*, ACM Press. 13 - 22.

OUYANG, T. AND R. DAVIS. 2009. Learning from Neighboring Strokes: Combining Appearance and Context for Multi-Domain Sketch Recognition. *Advances in Neural Information Processing Systems 22*: 1401-1409.

PAULSON, B. AND T. HAMMOND. 2008. PaleoSketch: Accurate Primitive Sketch Recognition and Beautification. *Intelligent User Interfaces (IUI '08)*, ACM Press. 1-10.

PETERSON, E., T. STAHOVICH, E. DOI AND C. ALVARADO. 2010. Grouping Strokes into Shapes in Hand-Drawn Diagrams. *Proc. of the 24th AAAI Conference on Artificial Intelligence (AAAI-10)*. 974-979.

PLIMMER, B., H. PURCHASE AND H. Y. YANG. 2010. SketchNode: Intelligent sketching support and formal diagramming. *OZCHI 2010*, ACM. 136-143.

RUBINE, D. H. 1991. Specifying gestures by example. *Proceedings of Siggraph '91*, ACM. 329-337.

SAUND, E., D. FLEET, D. LARNER AND J. MAHONEY. 2003. Perceptually-Supported Image Editing of Text and Graphics. *User Interface Software and Technology (UIST '03)*, ACM. pp. 183-192.

SAUND, E. AND T. P. MORAN. 1994. A perceptually-supported sketch editor. *Proceedings of the 7th annual ACM symposium on User interface software and technology*, ACM. 175-184.

SAUND;, E., J. MAHONEY;, D. FLEET;, D. LARNER; AND E. LANK. 2002. Perceptual Organization as a Foundation for Intelligent Sketch Editing. *AAAI Spring Symposium on Sketch Understanding*, American Association for Artificial Intelligence. 118-125.

SHILMAN, M. AND P. VIOLA. 2004. Spatial recognition and grouping of text and graphics. *Eurographics Workshop on Sketch-Based Interfaces and Modeling*. 91-95.

SHILMAN, M., Z. WEI;, S. RAGHUPATHY;, P. SIMARD AND D. JONES. 2003. Discerning structure from freeform handwritten notes. *Document Analysis and Recognition*. 60 - 65.

WOBBROCK, J. O., A. D. WILSON AND Y. LI. 2007. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. *User interface software and technology*, ACM. 159-168.