Chart Image Classification Using Multiple-Instance Learning

Weihua Huang, Siqi Zong, Chew Lim Tan
School of Computing, National University of Singapore
3 Science Drive 2, Singapore 117543
{huangwh, zongsiqi, tancl}@comp.nus.edu.sg

Abstract

An important step in chart image understanding is to identify the type of the input image so that corresponding interpretation can be performed. In this paper, we model the chart image classification as a multiple-instance learning problem. A chart image is treated as a bag containing a set of instances that are graphical symbols. For both training and recognition, shape detection is performed and general shape descriptors are used to form feature vectors. For the training images, the correlation factor (CF) of each shape is calculated for each chart type. The learnt CFs are then used to estimate the type of a new input image. Comparing with traditional multiple-instance learning algorithms, we allow negative examples to be less restrictive and hence easier to provide. Using our method, both the type and the data components of the chart image can be obtained in one-pass. The experimental results show that our approach works reasonably well.

1. Introduction

Chart is one of the most commonly used types of infographics for presenting data etc. Chart recognition and understanding is a relatively young research field and is attracting more and more research interests in recent years. Some of the reported works directly deal with electronic charts [1-2]. On the other hand, some works deal with charts that are converted into raster images [3-6]. Through our study of the literature, we find out that there are two common drawbacks in the existing approaches. First of all, most of the methods make assumption on the availability of chart type information so that predefined structural models and constraints can be applied. Secondly, although there was a work on chart image classification proposed in [4], the method relies on low-level features, such as foreground/background transition, that are not useful for high-level interpretation. To interpret the content of the chart, the whole image needs to be re-processed again. Putting the problem into the context of document image recognition, we suggest that the complete schema of chart recognition should contain the three main steps shown in Figure 1. The efficiency of the system can be guaranteed if output of early steps becomes useful input to later steps.

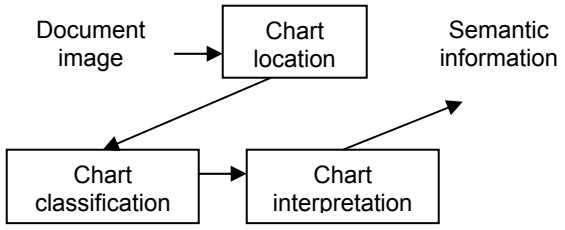


Figure 1. Chart image recognition schema

In this paper, we propose a novel approach that performs chart image classification and data component detection in one-pass based on machine learning. Data components are referred to as the graphical components in a chart that represent data. In a chart type, data always have a homogeneous representation and the representation is often shape-based graphical symbols. Furthermore, different chart types represent data using different shape (or combination of shapes). Based on the correlations among shapes, data representations and chart types, we suggest that the chart classification problem can be modeled as a multiple-instance learning problem. The result of learning helps to answer two questions: which shape (or combination of shapes) represents data in a chart? How closely is a shape (or combination of shapes) correlated to the chart type? The answer to the first question can be used for further interpretation of the chart content and the answer to the second question is for chart image classification.

Our approach is a modified version of the *Diverse Density* algorithm for solving multiple-instance learning problems [7]. The feature vector used for learning and matching is obtained from general shapes

extracted from training images. We define the degree of correlation between a shape (or combination of shapes) and a chart type as the correlation factor (CF). The task of the learning process is to identify the shapes appearing in every positive example with high CF. Figure 2 shows the detailed steps in the proposed approach. The method proposed works as a key step in a chart image understanding system being developed.

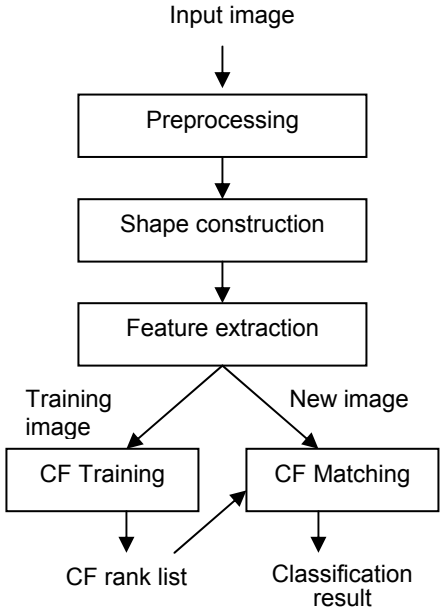


Figure 2. Steps in the proposed method

The remaining sections of the paper are organized as follows. Section 2 further discusses the motivation of applying multiple-instance learning in our approach. Section 3 introduces the proposed approach in details. Section 4 presents the results of experiments and some discussions about them. Section 5 concludes the paper by summarizing the contributions and future work.

2. Why multiple-instance learning?

The first design issue is that whether a learning approach should be applied. Most graphical symbol recognition methods fall into two categories: recognition with a known database of reference symbols, or recognition with a learning phase [8]. The choice is mainly based on two factors: whether the graphical symbols can be exhaustively listed and whether the system is expected to be self-extensible to new types of input. The system we are developing aims for handling general types of charts, and is expected to be extended to handle other types of infographics. Thus it should not make assumption on what kind of graphical symbols may appear, and a learning based approach will be more appropriate here.

Another concern is that symbol recognition usually requires preliminary segmentation phase [8], which is also true in our case. As we mentioned in section 1, most graphical symbols used by charts are shape-based. Thus shape-level segmentation needs to be performed. Error propagation may occur during *segmentation followed by recognition*. The situation is less severe, however, when the inputs to be handled are schemas or diagrams consisting of mainly symbols and connection lines. As chart is a form of diagrams containing well structured graphical symbols, segmentation errors can be controlled at a reasonably low level.

If shapes can be properly segmented, the most straightforward approach is to use a supervised learning, labeling every shape in the training examples. But it requires significant amount of human computer interaction and a good way of shape-level labeling, which increases the complexity of the system. If we treat the shapes as instances and the input images as bags, as we mentioned in the introduction, then a better alternative is to use multiple-instance learning, a variation of supervised learning.

Multiple-instance learning problem has been studied by a number of researchers and its application in the computer vision field includes image indexing, retrieval and classification etc. The most popular algorithm used is the *Diverse Density* algorithm [7]. Lakshmi Ratan et al proposed a framework to learn query concepts from training images based on the *Diverse Density* algorithm [9, 10]. The learned "concepts" are simple templates that capture the color, texture and spatial properties of the class of images. Cheng Yang et al further improve Lakshmi's framework to retrieve image from a database of natural images [11]. In their approach, feature vectors are formed instead of templates, and similarity measure between feature vectors of the query image and database images is calculated. Comparing to Lakshmi's framework, the obvious advantage is that no pre-defined knowledge about what object to look for is needed. This advantage is sustained in our system in the way that no predefined symbol is required and the feature vector of every shape that appears is examined.

3. Details of the proposed approach

3.1. The modified *Diverse Density* algorithm

The *Diverse Density* algorithm was proposed aiming at solving the multiple-instance learning

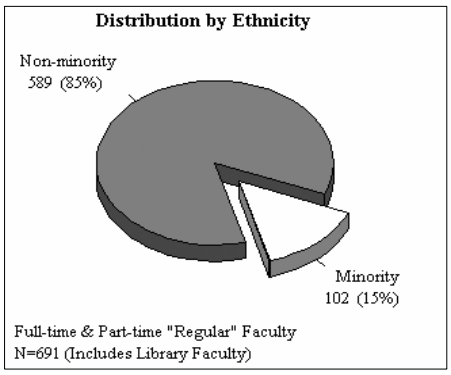
problem. In a traditional multiple-instance learning problem, input vectors ($x_{i1}, x_{i2}, \ldots, x_{in}$) (called *instances*) are grouped together to form a *bag*, and they are collectively labeled with a *y* value of TRUE (*positive bag*) or FALSE (*negative bag*). The instances corresponding to *y*=TRUE are called *positive instances*, and the instances corresponding to *y*=FALSE are called *negative instances*. The *Diverse Density* algorithm requires that at least one of the instances in a positive bag must correspond to *y*=TRUE, and all of the instances in a negative bag must correspond to *y*=FALSE. In other words, a negative bag should not contain any positive instances. When applied to chart image classification, this requirement for the negative bag is too strict and may cause difficulty for the user to provide such cases as charts often contain a number of different shapes. For example, a pie chart may also contain rectangles and squares to illustrate legends etc. Further more, the algorithm looks for a single point in the feature space where the *Diverse Density* is maximal. So another limitation is that there might be more than one instance that has high *Diverse Density* value, but only one of them is selected and others are ignored. This is also not desirable in our case, as symbols representing data components may consist of more than one shape thus all these shapes should be considered as positive instances.

To overcome these two limitations to better tackle the problem we are facing, the original *Diverse Density* algorithm is modified. The bags, instances and positive bags are defined in the same way but the definition of negative bags is more lenient. A selected negative example may contain positive instances, which makes the task easier for the user. Furthermore, we calculate the correlation factor (CF) for every instance and generate a rank list for matching instead of just choosing the one with the maximum CF.

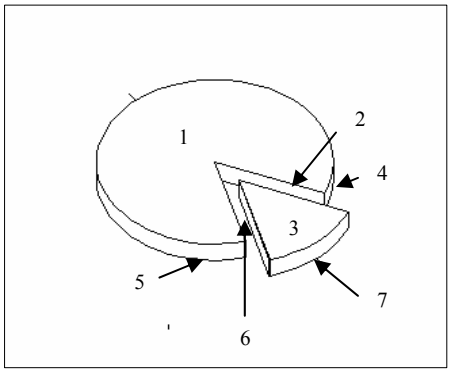
3.2. Preprocessing and feature extraction

In preprocessing, each input image is converted to a grayscale image and several steps are carried out:

- Text is removed from the image through connected component analysis [12]. Also in this step, the small noise components are removed.
- Edge map is obtained through edge detection.
- Vectorization is performed to convert image edges to a set of lines and arcs in the vector form.
- Based on the vectorized lines and arcs, a graph G(V, E) is formed where V is the set of intersection points among the lines and arcs, and E is the set of segments (either straight line segment or arc segment) between intersection points. Shape construction is a process of finding the Minimum Cycle Basis (MCB) [13] on the graph G, and an efficient algorithm was proposed by Alfredo Ferreira Jr. et al [14]. However the original algorithm only finds polygons from a set of straight lines, thus some of the steps are modified to take care of arc segments. Figure 3 shows an example of graph constructed from a 3D pie chart, from which 7 shapes are constructed.



(a) The original image



(b) Basic shapes obtained.

Figure 3. Example of shape construction from an input image. There are seven shapes in (b) labeled with numbers

Now a collection of shapes are obtained. The edges in a shape are classified into three types: (1) straight line, (2) circular arc or (3) elliptic arc. Although these three types of edges are not sufficient for general shapes, they can cover all the edges extracted from the chart images we examined since shapes in chart images are relatively more regular. Four shape descriptors are used to form the feature vector for each shape constructed: number of edges n_i for each edge type i; order o among the edges (represented as a sequence of edges); number of parallel edge pairs n_p; number of symmetric axes n_s. Thus a feature vector can be represented as $<n_1, n_2, n_3, o, n_p, n_s>$. We choose these

four shape descriptors because they are all invariant to translation, rotation and scaling, and a combination of them can uniquely define a shape class.

3.3. The training process

To train a chart type A, a set of positive bags $B^+ = \{B_j^+, j = 1,2,...,n\}$ and a set of negative bags $B^- = \{B_k^-, k = 1,2,...,m\}$ are provided by the user. Each bag is an image containing shapes. The first step is to find out the universal set of components (shapes) $C = \{C_i, i = 1, 2, ..., l\}$ where $C = B^+ \cup B^-$. Then the correlation factor $CF(C_i, A)$ between each C_i and a chart type A is derived from the conditional probability $P(C_i \mid B^+, B^-)$. Assuming that the training examples are conditionally independent given the chart type A, and by applying Bayes' rule (assuming an uninformative prior over the shape C_i), we can get:

$$P(C_i \mid B^+, B^-) = \prod_{j=1}^{n} P(C_i \mid B_j^+) \prod_{k=1}^{m} P(C_i \mid B_k^-) / P(C_i)^{n+m-1} \quad (1)$$

where

$$P(C_i \mid B_j^+) = 1 - \exp(-Num(Ci, B_j^+)) \quad (2)$$

$$P(C_i \mid B_k^-) = \exp(-Num(Ci, B_k^-)) \quad (3)$$

As $P(C_i)$ is independent of the training examples, we take it out from the expression in (1) to get:

$$CF(C_i, A) = \prod_{j=1}^{n} P(C_i \mid B_j^+) \prod_{k=1}^{m} P(C_i \mid B_k^-) \quad (4)$$

The derivation of formula (1) can be found in [7]. Both $P(C_i \mid B_j^+)$ and $P(C_i \mid B_k^-)$ are exponential functions that depend on the number of shapes matching the component C_i. $Num(C_i, B_j^+)$ calculates the number of shapes that match component C_i in the positive example B^+_j. As $Num(C_i, B^+_j)$ increases, $P(C_i \mid B^+_j)$ increases and is approaching 1. $Num(C_i, B_k^-)$ calculates the number of shapes that match component C_i in the negative example B_k^-. As $Num(C_i, B_k^-)$ increases, $P(C_i \mid B_k^-)$ decreases and is approaching 0. Exact matching between feature vectors is required.

3.4. The matching process

For a new image that is also treated as a bag containing a number of components C' = $\{C_g, g = 1, 2, ..., h\}$, we calculate the similarity between C' and type A as:

$$Sim(A, C') = \sum_{g=1}^{h} Num(C_g, C') CF(C_g, A) \quad (5)$$

where $Num(C_g, C')$ counts the number of occurrences of component C_g in the given image C'. If $C_g \in C$, then $CF(C_g, A)$ is pre-computed during the training process, otherwise $CF(C_g, A) = 0$. Similarity between the new image and every existing chart type is calculated, and the chart type that results in the highest similarity value is deemed to be the type of the new image. It is also possible that the new image belongs to a new chart type that was not presented during the training process. Thus a cut-off value can be set to judge whether the new images belongs to any of the existing types.

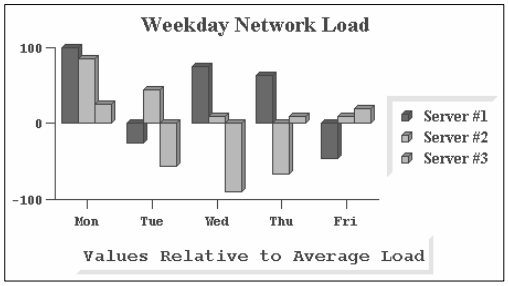


Figure 4. A 3D bar chart

3.5. Detecting combination of shapes

In some cases, the data components in a chart are represented using more complex graphical symbols that are combinations of shapes. This happens very often for 3D chart types. Figure 4 shows an example of 3D bar chart in which the data components are represented as cuboids that consist of 1 rectangle and 2 parallelograms. Based on the observation of homogenous representation of data in charts, the shapes that form a data component should have a high degree of co-occurrence in the set of shapes recognized. Another heuristic is that the shapes forming a single symbol are not separated, which means they are neighbors of each other. Thus an extra step to detect a complex symbol (data component) is to identify those shapes that are neighbors of each other and have high degree of co-occurrence. Two shapes are neighbors if they share a common edge or part of an edge.

To find out the probability of a combination of shapes being a complex symbol, we can compute the

degree of neighborhood *DoN* between two shape types T_1 and T_2. If the number of T_1 shapes in a given image is N_1 and the number of T_2 shapes in the same image is N_2, we can find out the number of T_1-T_2 neighboring pairs $N_{neighbor}$. Then *DoN* is calculated as:

$$DoN = \left(\frac{N_{neighbor}}{N_1} + \frac{N_{neighbor}}{N_2} \right) / 2 \qquad (6)$$

Note that the value of *DoN* falls in interval [0, 1]. When none of the T_1 shapes is neighbor of T_2 shapes, $N_{neighbor}$ becomes 0 and thus *DoN* becomes 0. In this situation, the two types of shape never appear together as neighbors. When all T_1 shapes are neighbors of T_2 shapes, and $N_1 = N_2$, *DoN* reaches its maximum value of 1. In this situation, the two types of shape always appear together as neighbor of each other. Thus we can see that the higher the *DoN* is, the more possible that the two types of shapes form a complex symbol.

4. Experimental results and discussions

For testing purpose, we collected a set of 210 chart images that were either taken from the internet or scanned in. The chart types and number of charts in each type are shown in Table 1.

Table 1. Images in the testing data

Type	Number of images
2D bar chart	80
2D pie chart	48
3D pie chart	12
Line chart	60
Doughnut chart	10
Total	210

The scanned images are relatively noisier than the downloaded images, with blur edges, small noise components and skew angles etc. Figure 5 shows an example of scanned images.

The experiment was carried out in 20 test runs. During each test run, a number of images were randomly chosen from each chart type to form the set of training images I_{train} and the remaining images became the testing images I_{test}. During training process, one chart type was learnt at a time and the CF values were stored. During matching, formula (5) was applied and the chart type returning the highest similarity value was assign to the testing image. Due to the space limit, only the average accuracy of chart classification for the 20 runs is presented in Table 2. In each test run, the accuracy is calculated as the percentage of testing images that were correctly classified.

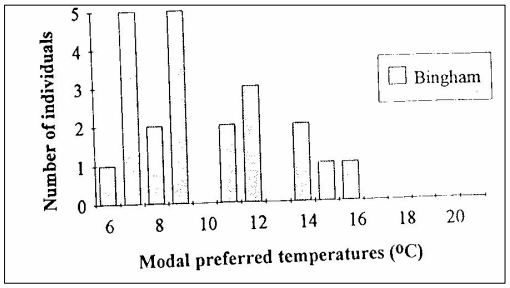


Figure 5. Example of scanned chart images

Table 2. Summary of classification results

No. of I_{train} per run	Type	No. of I_{test} per run	Average Accuracy (%)
3	2D bar	77	88.81
	2D pie	45	89.33
	3D pie	9	91.11
	Line	57	14.04
	Doughnut	7	100
5	2D bar	75	95.00
	2D pie	43	89.19
	3D pie	7	95.71
	Line	55	3.91
	Doughnut	5	100

From Table 2, we can see that the accuracy of chart image classification is very good for all chart types except line chart. This is because of the assumption we made in the introduction section that data are represented using shapes, which is not true for line chart where data are actually represented using x-y plots. Although some shapes appear occasionally, none of them is closely correlated to the type. The correlation factor of most shapes for line chart is zero, and as a result, the similarity value calculated is also zero, causing the system to fail to recognize the correct type for line chart images. When all the similarity values are too low for an input image, the type of the image will be "unidentified".

During the training process, the system also identified the shapes with the highest CF value. These shapes are the best candidates to be the representation of data component for each chart type, and they are summarized in Table 3. The first three values in the feature vector show the number of edges for each edge type. For example, a data component in 2D bar chart has 4 straight line edges, 0 circular arc edges and 0 elliptic arc edges; while a data component in 2D pie chart has 2 straight line edges and 1 circular arc edge. The fourth value in the feature vector is a sequence among edges reflecting how the edges are ordered (denoting a straight line as 1, a circular arc as 2 and an elliptic arc as 3). Rotation is taken care of here, thus

the order 131 is the same as 311. The last two values in the feature vector show the number of parallel edges and number of symmetric axes.

Table 3. Data component identified for chart types

Chart type	Feature vector	Sample shape
2D bar	<4,0,0, 1111, 2, 2>	
2D pie	<2,1,0, 121, 0, 1>	
3D pie	<2,0,1, 131, 0, 0>	
Doughnut	<2,2,0, 1212, 1, 1>	

Another output of the system is the degree of neighborhood among shapes, calculated by formula (6). One restriction is that the shapes to be considered must have non-zero CF value, meaning that they must appear in all positive examples. With this restriction, we only found one combination of shapes whose *DoN* > 0 for all test runs: <2,0,1, 131, 0, 0> and <2,0,2, 1331, 2, 0>. This is a typical combination of shapes in 3D pie charts, such as shape no.1 and no. 5, or shape no. 3 and no. 7 in Figure 3(b). For the 2D charts, no common combination of shapes was found. This is expected, since data are represented using single shape in these 2D charts.

5. Conclusion

This paper presents a novel work of chart image classification based on multiple-instance learning. Our approach does not require pre-defined shape templates, instead general shape descriptors are used as feature vectors. Unlike traditional multiple-instance learning algorithm, our algorithm does not require that the negative examples contain no positive instances corresponding to the class to be learnt, thus the training is a lot easier for the user. Furthermore, we maintain the correlation factor between the instances and all chart types instead of just the maximal one. Finally the shapes learnt can also be re-used for further interpretation of chart contents.

In the future, shape inheritance can be investigated to allow more tolerance during feature matching. Also, the feature vectors can be further extended to include other features that are not shape-based. Then our approach is expected to be more generally applicable and have more classification power.

Acknowledgment: This research is supported by A*STAR under grant no. 042 101 0085.

References

[1] S. Elzer, S. Carberry, I. Zukerman, D. Chester, N. Green and S. Demir, “A Probabilistic Framework for Recognizing Intention in Information Graphics”, *IJCAI 2005*, pages 1042-1047, 2005.

[2] S. Carberry, S. Elzer, N. Green, K. McCoy and D. Chester, “Understanding Information Graphics: A Discourse-Level Problem”, *Proceedings of SIGDial*, pp. 1-12, 2003.

[3] N. Yokokura and T. Watanabe, “Layout-Based Approach for extracting constructive elements of bar-charts”, *GREC'97*, pp163-174.

[4] Y. P. Zhou and C. L. Tan, “Learning-based scientific chart recognition”, *GREC2001*, page 482-492, 2001.

[5] Y. P. Zhou and C. L. Tan, “Hough technique for bar charts detection and recognition in document images”, *ICIP 2000*, page 494-497, 2000.

[6] W. H. Huang, C. L. Tan and W. K. Leow, “Model based chart image recognition”, *Int. Workshop on Graphics Recognition, GREC2003*, Barcelona, 2003,.

[7] O. Maron, T. L. P´erez, “A framework for multiple-instance learning”, *Advances in Neural Information Processing Systems*, vol. 10, pp. 570-576, 1997.

[8] K. Tombre and B. Lamiroy, “Graphics recognition - From re-engineering to retrieval”, *Proc. of 7th ICDAR,* Edinburgh (Scotland, UK)*,* pp. 148--155, August 2003.

[9] O. Maron and A. L. Ratan, “Multiple-instance learning for natural scene classification”, *Proc. 15th Int. Conf. on Machine Learning*, page 341-349, 1998.

[10] A. L. Ratan, O. Maron, W. E. L. Grimson and T. L. P´erez, “A framework for learning query concepts in image classification”, *CVPR,* 1999, pp. 423-429.

[11] C. Yang and T. L. Perez, “Image Database Retrieval with Multiple-Instance Learning Techniques”, *International Conference on Data Engineering, ICDE*, page 233-243, 2000.

[12] K. Tombre, S. Tabbone, L. Pélissier, B. Lamiroy, and P. Dosch, “Text/Graphics Separation Revisited”, *5th Intl. Workshop on DAS (2002)*, page 200-211, 2002.

[13] M. M. Syslo, “An efficient cycle vector space algorithm for listing all cycles of a planar graph”, *SIAM Journal on Computing*, vol. 10 no. 4, page 797-808, 1981.

[14] A. Ferreira and M. Fonseca and J. Jorge, “Polygon Detection from a Set of Lines”, *Proceedings of 12 Encontro Portugu es de Computacao Gr afica (12th EPCG)*, Porto, Portugal, 2003, pp. 159-162.