ELEMENT DETECTION IN JAPANESE COMIC BOOK PANELS

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Toshihiro Kuboi

August 2014

COMMITTEE MEMBERSHIP

TITLE:	Element Detection in Japanese Comic Book Panels
AUTHOR:	Toshihiro Kuboi
DATE SUBMITTED:	August 2014
COMMITTEE CHAIR:	Foaad Khosmood, Ph.D. Assistant Professor of Computer Science
COMMITTEE MEMBER:	Franz Kurfess, Ph.D. Professor of Computer Science
COMMITTEE MEMBER:	John Seng, Ph.D. Associate Professor of Computer Science

ABSTRACT

Element Detection in Japanese Comic Book Panels

Toshihiro Kuboi

Comic books are a unique and increasingly popular form of entertainment combining visual and textual elements of communication. This work pertains to making comic books more accessible. Specifically, this paper explains how we detect elements such as speech bubbles present in Japanese comic book panels. Some applications of the work presented in this paper are automatic detection of text and its transformation into audio or into other languages. Automatic detection of elements can also allow reasoning and analysis at a deeper semantic level than what's possible today. Our approach uses an expert system and a machine learning system. The expert system process information from images and inspires feature sets which help train the machine learning system. The expert system detects speech bubbles based on heuristics. The machine learning system uses machine learning algorithms. Specifically, Naive Bayes, Maximum Entropy, and support vector machine are used to detect speech bubbles. The algorithms are trained in a fully-supervised way and a semi-supervised way. Both the expert system and the machine learning system achieved high accuracy. We are able to train the machine learning algorithms to detect speech bubbles just as accurately as the expert system. We also applied the same approach to eye detection of characters in the panels, and are able to detect majority of the eyes but with low precision. However, we are able to improve the performance of our eye detection system significantly by combining the SVM and either the Naive Bayes or the AdaBoost classifiers.

Keywords: Machine Learning, Computer Vision, Artificial Intelligence

ACKNOWLEDGMENTS

This thesis is inspired by Eriq Augustine's original project on the machine translation of Japanese comic books. I would like to thank Eriq for his guidance and assistance on the selection of this thesis topic.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. Introduction

Many books have been published since the invention of the printing press by Gutenberg. Google has computed that there are 129,864,880 books in the entire world (Parr, 2010). Even though the current trend is tipping toward publishing using electronic media instead of paper based media, a large number of books is still being published today (Bowker, 2014). Books are perhaps easier to read and engage for us humans than electronic media, but they are more difficult for computers to work with. Being able to process information contained in books with computers is very important because there is a tremendous need for cataloging books, searching information contained in them, and doing further processing, such as automatic translation from one language into another. According to UNESCO, 164,499 books have been translated into English from other languages from 1979 to 2012 (UNESCO, n.d.). A well-known example is a Google's project in which huge volumes of books are scanned and transformed into electronic form so that their contents can be made accessible online (Parr, 2010).

What we present in this thesis pertains to making comic books more accessible online. More specifically, this thesis attempts to detect elements such as speech bubbles present in scanned images of pages from Japanese comic books. A scanned image of a page of a book such as a comic book or a picture book for children has multiple elements. For instance Japanese comic books, called Manga in Japanese, have speech bubbles containing the text of characters' verbal utterances. One application of detection of the elements in a scanned image of a page from a comic book is automatic detection of text and its translation from Japanese to English. Japanese comic books have gained popularity in the countries outside Japan, and there is a tremendous need for translating

Japanese text in the comic books into English. The first step of detecting speech text in a comic book would be detecting speech bubbles on a page.

2. Background

2.1. Speech Bubbles

Pages in comic books have multiple elements including text and pictures. Text in comic books is typically enclosed in speech bubbles. A speech bubble usually has an oval or circular shape (bubble) with a narrow elongated part sticking out of the bubble, pointing toward a character in the comic, indicating that the text of speech is uttered by that character.

2.2. Object / Element detection

In the field of computer vision, object detection is a task to detect objects present in a given image. For example, an object detection technique attempts to determine whether a person is present in a given picture or video frame. In order to detect objects present in a given image, the image needs to be segmented into regions to identify the region of interest. The task of image segmentation can be done by a technique called connected component analysis, in which neighboring pixels which share similar features are identified and grouped together as a region (Horn, 1986, pp. 65-89). Then, the region is determined whether or not it is the object of interest.

We call our task element detection instead of object detection because our goal is to detect elements such as speech bubbles, which may not be considered objects because they consist of text. We use elements as a term that includes text as well as objects. In other words, we use “element” as a term broader than “object”.

2.3. Object Recognition

Object recognition is a task to recognize specific instances of objects in a given image. For example, an object recognition technique attempts to identify whether a person present in a given image is person A or B. Again, we would call our recognition task as “element recognition” instead of “object recognition” because our goal would be to recognize elements in a given image which are not necessarily considered as objects.

2.4. Pre-Process and Post-Process

In computer vision, normally an image of interest is pre-processed before the main task of interest. The pre-processing typically involves filtering to remove noise. The image is also post-processed after the main task to improve the result. The post-processing includes, for example, visualization of the results by annotating the original image, and the application of computationally expensive operations to exclude false positives.

3. Problem Statement / Objectives

Our objectives are two fold:

1) Build a system that can detect the elements very accurately, taking advantage of a human expert.

2) Use machine learning algorithms and train them to detect the elements very accurately, without the help of the human expert.

In this paper, we focus on detecting speech bubbles and eyes of characters.

4. Motivation

Being able to process information contained in books with computers is very important because there is a tremendous need for cataloging books, searching information contained in them, and doing further processing, such as automatic translation from one language into another. Japanese comic books have gained popularity in the countries outside Japan, and there is a tremendous need for translating Japanese text in the comic books into English. The first step of detecting speech text in a comic book would be detecting speech bubbles in a page. The other application of our work would be automatic narration and a search for elements contained in images. There is a merit on speech bubbles, as opposed to just text, because a speech bubble indicates which character in the image made the speech. Thus, we can transform the story depicted in the image into text or audio format.

5. Related Work

5.1. Document Image Analysis

Wong, Casey, and Wahl (1982) proposed a method to identify regions containing text within a document image by using run length smoothing algorithm (RLSA). The basic RLSA is applied to a binary sequence in which white pixels are represented by 0's and black pixels by 1's. The RLSA transforms 0's in the binary sequence into 1's if the number of adjacent 0's is less than or equal to a predefined number N. This transformation has the effect of linking together neighboring black pixels that are separated by less than N pixels. The RLSA smoothing is applied both horizontally and vertically. With an appropriate choice of N, the linked areas will be regions of pixels with similar characteristics. The degree of the linkage depends on the value of N (Wong et al., 1982).

Fisher, Hinds, and D'Amato (1990) proposed a rule-based system for automatically segmenting a document image into regions of text and non-text. The image is initially enhanced by techniques such as adaptive thresholding, morphological processing, and skew detection and correction. The image is smoothed by RLSA and segmented based on statistics of the connected components.

Tombre, Tabbone, Pélissier, Lamiroy, and Dosch (2002) made an improvement of the method proposed in (Fisher et al., 1990) by choosing the right thresholds. Tombre et al. (2002) also proposed a post-processing step for retrieving text components touching the graphics, by extending components recognized as text to graphics components.

Fletcher and Kasturi (1988) presented development and implementation of an algorithm for automated text string separation. The algorithm is relatively independent of changes in text font style and size, and of string orientation. The algorithm generates connected components and applies the Hough transform in order to group together components into logical character strings. Then the strings are separated from the graphics. The components larger than threshold are discarded. The Hough transform is an algorithm which finds lines associated with points in the image. The connected components are grouped into strings associated with a particular line, and ordered according to their distance along the line (Fletcher & Kasturi, 1988).

Peng, Chi, Siu, and Feng (2000) proposed a document processing engine, which analyzed and decomposed the optical image of a paper document into a series of component blocks. The component blocks are encoded and stored in a structured and compact format.

Peng, Long, and Chi (2003) proposed a document image recognition system. The system uses template matching based on Component Block Projections (CBP), which are the concatenated directional projection vectors of the component blocks of a document image. CBP-based template-matching methods achieved a very high matching accuracy even for a large template set and significantly distorted input images.

Hu, Quinn, Rose, Bederson, and Arisaka (2008) presented a system that enhanced the readability of scanned picture books. The system separates textual from visual content and also decreases the size of the image. The text is made easy to read by being displayed as computer-generated text instead of an image. Their algorithm uses color thresholding, connected component analysis and morphological transformation. This

algorithm is based on two assumptions: 1) Text is darker than the background; and 2) within a page, the font is homogeneous. A connected component is considered text only if it is not significantly wider and not taller than a word.

5.2. Sketched Object Category Recognition

Eitz, Hays, and Alexa (2012) proposed an approach to recognize a category of objects a given sketch belongs to. A sketch is represented by a large number of local features encoding local orientation estimates. The features are extracted by computing orientations of edges in the sketch and binning them into a histogram. Then, the features are grouped into clusters or “visual words” using k-means clustering. As a result, a sketch is represented as a frequency histogram of visual words. They built 250 binary SVM classifiers, and each classifier is trained to classify one category against the rest of the categories in order to classify sketches into 250 categories they identified. With the combined classifiers, they achieved 56% classification accuracy.

5.3. Face Detection

Viola and Jones (2004) developed a robust real-time face detection algorithm. They used a cascade of classifiers which are trained in sequence. The cascaded classifiers reject many of the negative sub-windows as samples advance through the cascade. The classifiers in the cascade are trained by Adaptive Boosting (AdaBoost) (Freund & Schapire, 1995) boosting algorithm. The AdaBoost combines multiple classifiers and boost the performance of the classifiers by dynamically adjusting the weights of training samples based on the classification errors. Their work focused on detecting faces of people in photographic images and the images are segmented by sliding windows across it. The rectangular features, which are created by subtracting the sum of the pixel

values of one or more rectangular regions from those of other rectangular regions, are used.

6. Approach

Our approach is to use a heuristic-based expert system and machine learning for detecting elements present in images. The image processing system serves as a pre-processing system that extracts features that will be fed to both the expert system and the machine learning system. Specifically, the image processing system segments an image into regions and extracts regions that are likely to contain elements of interest. Then the features are extracted from the regions. The regions are classified whether or not they are elements of interest by the expert system based on heuristics. The expert system serves as our base line.

Our ultimate goal is to let a machine learn by itself to detect elements. With the expert system, whenever new samples are encountered heuristics or parameters need to be adjusted by a programmer. With the machine learning system, the system adjusts itself to new samples. In our approach, we train the machine learning system on the same samples given to the expert system. Then we let the machine learning system classify unseen samples.

We use two types of machine learning. One is supervised learning. In supervised learning, the system is supervised by humans. All the training data is labeled as either positive or negative examples by humans manually. The supervised learning is good when a lot of hand-labeled training data is available (Liu, 2011, pp. 63-129).

The other type of machine learning approach we use is semi-supervised learning. In semi-supervised learning, not all the training data used is hand labeled. Some of the training data is produced by the system itself. In our approach, un-seen data labeled by the trained machine learning system is added to the training data. Semi-supervised

learning is useful when a lot of hand-labeled training data is not available. In our approach, the machine learning algorithms are trained with training data labeled by the machine learning algorithms themselves (Liu, 2011, pp. 171-206).

Figure 1 shows the overview of our element detection pipeline. It mainly consists of the image processing system, the expert system, and the machine learning system.

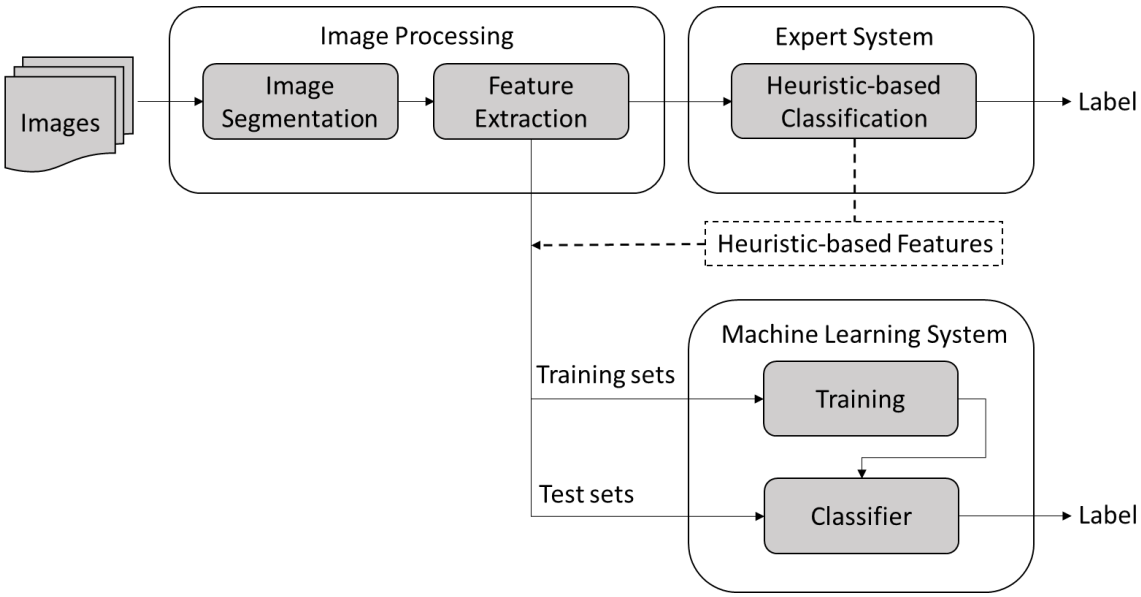

Figure 1. Overview of our element detection pipeline

6.1. Image Processing

The image processing system imports image files and process the images.

6.1.1. Pre-Processing

The whole process begins as an image is loaded into our image processing system. Specifically, a jpeg image is converted to an array of pixel values. The value of a pixel is

represented by three bytes, each byte representing one of the three channels of RGB. For this task, a library called ImageMagick is used.

The image is a gray scale image and consists of one frame of a comic book page. The image is first filtered with a median filter (Gonzalez & Woods, 2002, pp. 123-124). Specifically, for a pixel, the median value of the colors of its eight neighboring pixels is selected as its color. The median filter is good for removing noises caused by dust or dirty glass in the scanner used. The filter is also good at preserving edges.

Although the image is a gray scale image, it is imported to the system as a color image. Thus, it has three channels. The image is flattened as a black and white image by thresholding: pixels whose colors are lighter than the threshold value are converted to white; otherwise, they are converted to black. Then, the image is smoothed with a Gaussian filter to remove noise.

The image is further smoothed by the Run Length Smoothing Algorithm (RLSA) (Wong, Casey, & Wahl, 1982). At this point the image is basically represented by arrays of pixels whose values are either 1 for white pixels or 0 for black pixels. The RLSA transforms 0's in the binary sequence into 1's if the number of adjacent 0's is less than or equal to a predefined number N (Wong et al., 1982). This transformation has the effect of linking together neighboring black pixels that are separated by less than N pixels (Wong et al., 1982). The RLSA smoothing is applied both horizontally and vertically. In our approach, the value of N used for the horizontal RLSA smoothing is 5. The value for the vertical smoothing is 30. The values are selected in empirical fashion so that non-text black pixels are connected together and the text black pixels are connected together. The

value for the vertical smoothing is much larger than that of the horizontal smoothing because Japanese text is written in a top-down fashion.

In image processing, edge detection is often used as a part of the preprocessing stage for object detection or recognition (Canny, 1986). This is because the presence of edges is a good indication of the presence of objects in that part of the image. For example, an object can be separated from its background by detecting its edges in a real-life image. Edges are extracted by looking for changes in the color values or intensity values of the pixels. The changes can be extracted by applying an operator that looks like one in Figure 3. The application of the operator approximates computation of derivatives of a curve which represents the value of the pixels (Gonzalez & Woods, 2002, pp. 572-580).

-1	0	1
-1	0	1
-1	0	1

Figure 2. An example of an operator for edge detection

For our task, we do not need to do the edge detection because the images we are dealing with are artificially drawn. In other words, the edges have been already provided by the authors of the comic books.

6.1.2. Image Segmentation

An image needs to be segmented because processing the entire image is inefficient. Some parts of the image may contain the elements of interest, but the rest of the image may not contain any. Therefore, parts of the image that are most likely to contain the

elements need to be segmented out for further processing. Usually, an image is segmented into parts which have similar characteristics such as color or intensity.

There are several ways to segment images. When an image is a black and white image, which is the case in our work, one way to segment the image is to segment it into black pixels and white pixels. A technique called the connected component analysis can be used for this task (Horn, 1986, pp. 65-89). In the connected component analysis, neighboring pixels (8 pixels in case of 8 connectivity) with similar characteristics are connected together into one component. The technique can be applied to images other than black and white images by using a threshold to categorize pixel into one group or the other. The analysis is usually done by scanning pixels from the top left corner to the bottom right corner.

The other technique segments an image by sliding a window of a certain size and by computing characteristics of pixels within the window. The sliding of the window may be done multiple times over the image with different sizes of the window.

In our approach, a technique called the region growing (Gonzalez & Woods, 2002, pp. 613-615) is used. The region growing is similar to the connected component analysis, but the analysis starts from a starting pixel and the connected component is grown outward from the pixel. For example, when segmenting a white region from a black region, for each white pixel encountered during the process, its 8 neighboring pixels are checked if they are white pixels as well. White pixels are incorporated into the region and the process is repeated for each newly added pixels. Thus, the region of white pixels grows outward in eight directions from the starting pixel. Figure 3 illustrates the mechanism of the region growing.

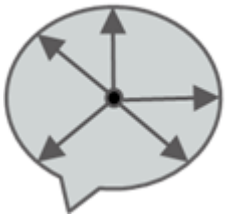

Figure 3. An illustration of the region growing

We use the region growing approach because it is efficient and suitable for the images we use. We are not interested in all the white pixels or all the black pixels. We are only interested in regions of white pixels which contain text pixels and are enclosed by black edges. Scanning each pixel in an image from the top left to the bottom right corner is not necessary. Moreover, the shape and the size of the speech bubbles vary and the region growing is more efficient than sliding windows of different sizes over the image.

In the image segmentation step, region growing is applied to the image to extract connected black pixels. Black pixels, which become starting points of region growing, are located by scanning pixels from the top left corner of the image to the bottom right corner. Therefore, the region growing can be easily replaced by the connected component analysis here. The connected black pixels that seem very likely to be text are selected and marked as candidate regions for further analysis.

The heuristic shown in Table 1 is used to determine whether or not the black pixels are text or not. The heuristics in the rows are concatenated with logical AND. In other words, groups of black pixels that meet all the conditions in the table are marked as possible text.

Table 1. Heuristic used for detecting text

	Heuristic for detecting text
1	70 < the number of the black pixels < 7999
2	horizontal range of the black pixels < 100
3	0.2 < the ratio of the horizontal range and the vertical range of the black pixels < 10
4	0.43 < The percentage of the area of the image the black pixels cover OR 3000 < the number of the black pixels

Figure 4 shows a result of the process described so far. All the text in the image in the original image on the left is segmented and assigned the same color (red). All the other connected black pixels are grouped as non-text components. Note that black pixels which are very close in the original image are connected together by the RLSA smoothing.

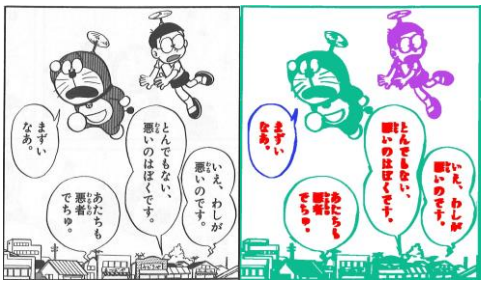

Figure 4. An example of segmentation of text

For each selected connected component, the image segmentation module analyzes whether or not the components are surrounded by black pixels. If the selected black pixel group is indeed text contained in speech bubbles, then they should be enclosed in speech bubbles. Whether or not a candidate region is enclosed by black pixels is

determined by the aforementioned region growing. A region consisting of a white pixel at the beginning is made to grow outward from the center of the region. When the center is a black pixel or the pixel in the center is not surrounded by more than 5 white pixels, a pixel within the bounding box of the black pixel group is randomly selected until a white pixel with the more than 5 surrounding white pixels are found. The black pixel group and the white pixels invaded by the process of the region growing are segmented out as one candidate region.

The region growing is terminated when no more neighboring white pixels that the region can grow into are found. This termination condition can be reached in two ways. One is when all the neighboring pixels are either already a part of the region or black pixels. Gaps among black pixels with equal to or less than 3 pixels are considered non-open pixels, and the growth process stops. The other condition is that the region has grown out of the bounding box of the black pixel group and a choke point has been reached. The choke point is a point where either the vertical or horizontal length of the region is less than 30 pixels and the length in the other direction is larger than a certain threshold. In other words, the choke point is where the region is very narrow in one direction but very wide in the other direction. The process of the region growing is made to stop at the choke point because if it is allowed to continue beyond the point, the region is very likely to grow out of the speech bubble.

6.1.3. Feature Extraction

Next, features are extracted from each candidate region. Table 2 shows the features extracted.

Table 2. Features extracted by the system

#	Features
1	The percentage of the area of the image the region occupies
2	The horizontal range of the region
3	The vertical range of the region
4	Whether or not the region is enclosed by black pixels
5	The percentage the region is enclosed by black pixels
6	Ratio between black pixels and white pixels
7	The distance between the center of the region and the center of the group of black pixels the region contains
8	The number of pixels
9	The histogram of the count of black pixels binned according to their horizontal positions
10	The histogram of the count of black pixels binned according to their vertical positions
11	The histogram of the count of black pixels surrounding the group of the black pixels marked as text binned according to their radial positions

The size of each region is approximated by the dimension of a rectangular box. The box's width and height corresponds to the maximum horizontal range and maximum vertical range of the region respectively: For example, if the positions of pixels in the X axis range between 10 and 100, the width or the horizontal range of the region is approximated to be 90, and if the positions in the Y axis range between 50 and 120, the height or the vertical range of the region is approximated to be 70. Then, the area of a region is approximated by multiplying the width and the height of the rectangular box.

Whether or not a region is enclosed by black pixels is determined by whether or not the region growing process terminates due to not being able to find space to grow into any farther. The region growing process is also terminated when it fails to terminate within a certain distance from the black pixels marked as text contained in the region. When the region is not enclosed by black pixels, the percentage of the region enclosed by black pixels is computed by dividing the number of white pixels adjacent to black pixels by the number of white pixels which are not adjacent to black pixels but on the edge of the region.

The distance between the center of the region and the center of the group of black pixels the region contains is obtained by computing the Euclidean distance between the centers of the rectangular boxes which define the boundaries of the region and the black pixels marked as text within the region.

The histograms of the count of black pixels are obtained by dividing the rectangular bounding box into 10 subregions along X axis (in case of feature #9) and Y axis (in case of feature #10) and counting the number of black pixels in each subregion.

The histogram of the count of black pixels surrounding the group of the black pixels marked as text are binned according to their radial positions (#11) is obtained as follows. Black pixels which lie along the radial lines of 0, 24, 48, 72, 96, 120, 144, 168, 192, 216, 240, 264, 288, 312, and 336 degree from the center of the rectangular bounding box of the region are counted and binned in the bin corresponding to the radial line.

6.2. Expert System

The expert system executes heuristics-based classification after the image processing stage, in which an image is segmented into regions and features are extracted from the regions.

Each candidate region is classified as either positive or negative, positive being a speech bubble. The heuristic used is shown in Table 3. The heuristic is basically whether or not the value of each feature falls within predetermined range, which are determined empirically by a human expert. All the rows of the heuristic are concatenated with logical AND. In other words, regions that satisfy all the rows of the conditions in Table 3 are classified as speech bubbles. Note that all the features listed in Table 2 are used except for the features #9-11. The features #9-11 are not used by the expert system because it can classify speech bubbles without them very accurately, but they are used by the machine learning system.

Table 3. Heuristics used for detecting speech bubbles

#	Heuristic
1	1% < The percentage of the area of the image the region occupies < 99%
2	The horizontal range of the region < 75% of the width of the image
3	The vertical range of the region < 75% of the height of the image
4, 5	Whether or not the region is enclosed by black pixels == true OR The percentage the region is enclosed by black pixels > 83%
6	Ratio between black pixels and white pixels > 0.059
7	The distance between the center of the region and the center of the group of black pixels the region contains < 57.4
8	The number of pixels > 6000

6.3. Machine Learning System

Three algorithms are used for the machine learning system. They are Naive Bayes, Maximum Entropy, and Support Vector Machines (SVM).

6.3.1.Naive Bayes

The Naive Bayes algorithm is based on Bayes rule and it assumes that each event occurs independently of the other events, hence it is called Naive Bayes (Liu, 2011, pp. 100-108). In the training stage, a Naive Bayes classifier accumulates support for all the events seen during the training. Among the events, the events we are interested in predicting are called classes. Other events are called features. Then the probability of each class and the probability of features given a class are computed. The trained classifier predicts a class when features are given by using the Bayes rule:

$$P(class|features) = \frac{P(class) * P(features|class)}{P(features)}$$

The classifier returns the class which maximizes the following equation:

$$\frac{P(features \mid class) * P(class)}{\Sigma\{P(features \mid class) * P(class)\} \text{ over all possible classes}}$$

(Bird, Klein, & Loper, 2009, pp. 246-250)(Liu, 2011, pp. 100-108)(NLTK Project, 2014)

We use Naive Bayes implementation included in Natural Language Tool Kit (NLTK) (NLTK Project, 2014).

6.3.2. Maximum Entropy

The Maximum Entropy classifier is a probabilistic classifier, but unlike the Naive Bayes classifier, Maximum Entropy does not assume that each event occurs independently of the other events. The classifier is based on the principle of Maximum Entropy. It selects a model which has the largest entropy from all the models that fit the training data. The principle of Maximum Entropy states that among the probability distributions that are known to us, the distribution with the largest entropy best represents the current state of knowledge (Bird et al., 2009).

The Maximum Entropy classifier is parameterized by a set of "weights", which are used to combine the joint-features that are generated from a featureset (fs) by an "encoding". In particular, the encoding maps each (fs, label) pair to a vector. The probability of each label is then computed using the following equation:

$$\text{prob(fs|label)} = \frac{\text{weights} \bullet \text{encode(fs,label)}}{\Sigma((\text{weights} \bullet \text{encode(fs,label)}) \text{ for l in labels})}$$

Where • is the dot product. (Bird et al., 2009, pp. 251-254)(NLTK Project, 2014)

We use the Maximum Entropy classifier with Improved Iterative Scaling (IIS) algorithm option included in NLTK.

6.3.3. Support Vector Machines

The Support Vector Machine (SVM) is a linear learning model, which is inherently a binary classifier. Given training data, the SVM tries to find a boundary that separates two classes present in the training data. The boundary can be a line, hyperplane, or any other shape. The SVM is sometimes called a maximum margin classifier because it tries to find a boundary between classes that gives the maximum margin between the classes. New examples are then mapped into the same space and predicted to belong to a category based on which side of the gap they fall on. In addition to performing linear

classification, the SVM can efficiently perform a non-linear classification using what is called the kernel trick, which implicitly maps inputs into high-dimensional feature spaces (Liu, 2011, pp. 109-124).

We use SVM implementation included in Waikato Environment for Knowledge Analysis (WEKA) (Hall, Frank, Holmes, Pfahringer, Reutemann, & Witten, 2009).

6.3.4. Features

The features used in the machine learning system are the features listed in Table 2 along with additional features. The additional features are whether or not each row of the conditions listed in Table 3 are satisfied. Therefore, there are 7 additional features which are inspired by the expert system and whose values are either 0 (in case a condition is not satisfied) or 1 (in case a condition is satisfied). In total, 58 features are used (of the 58 features, 10 correspond to the 10 bins of the histogram of the feature #9, another 10 correspond to the 10 bins of the histogram of the feature #10, and 15 correspond to the 15 bins of the histogram of the feature #11).

6.3.5. Training / Cross Validation

We experiment with two types of machine learning methodologies. One is supervised learning where all the training data is hand labeled. The other is semi-supervised learning where training data labeled by the trained machine learning system is used.

6.3.6. Supervised Learning

In the supervised learning, the machine learning algorithms are trained with training data which are manually labeled. The trained models are tested on cross validation data, which are also hand labeled. The ratio between the size of the training and that of the cross validation data is 2 : 1.

6.3.7. Semi-Supervised Learning

In the semi-supervised learning, the machine learning algorithms are trained with training data labeled by the supervised machine learning. After the training, the trained models are tested on unseen data and their performance is evaluated. The purpose of this experiment is to see how the machine learning algorithms perform in this set of conditions, and to see whether or not they can perform as well as the expert system.

7. Evaluation

7.1. Measurements of performance

The performance of a classification system is usually measured by its accuracy, precision, recall, and the F-score (Liu, 2011, pp. 63-132).

The accuracy measures the percentage of the agreement between the prediction and actual label of the test set. Thus accuracy is obtained by computing the following equation:

$$\text{Accuracy} = \frac{\text{\# of correct predictions}}{\text{Total \# of test sets}}$$

(Liu, 2011, p. 65)

It is sometimes very difficult to measure the performance of the system with accuracy alone. When the number of positive cases is very small in the test set, the system can produce very high accuracy by predicting all case as negatives. Computing the precision and the recall can provide more information about the performance of the system.

Precision measures the percentage of correctly predicted positive cases (true positives) over the total number of cases predicted as positive. In other words, it measures how accurate the system is when it predicted positive. Thus the precision is obtained by computing the following equation:

$$\text{Precision} = \frac{\text{\# of true positives}}{\text{\# of true positives} + \text{\# of false positives}}$$

(Liu, 2011, pp. 81-82)

Recall measures the percentage of the positive cases in the test sets correctly predicted by the system. Thus the recall is obtained by computing the following equation:

$$\text{Recall} = \frac{\text{\# of true positives}}{\text{\# of true positives} + \text{\# of false negatives}}$$

(Liu, 2011, pp. 81-82)

Precision and recall are somewhat mutually exclusive. A system that produces no false positives achieves a high precision. However, if it produces a lot of false negatives, it shows low recall. On the other hand, to achieve a high recall, a system usually needs to produce more false positives. Thus, improving one of the two measures may hurt the performance on the other measure. Therefore, it is difficult to compare two systems with different precisions and recalls.

In order to combine precision and recall and obtain one performance measure, the F-score is often used.

The F-score is obtained by computing the following equation:

$$\text{F-score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

In order for the F-score to be high, both the precision and the recall need to be high (Liu, 2011, p. 82).

We are going to evaluate the performance of the expert system and the machine learning system with the accuracy, precision, recall, and the F-score.

7.2. Data used

243 images from Doraemon, a very popular Japanese comic book series written by Fujiko-Fujio are used to evaluate the expert system and the machine learning system. Each image corresponds to one panel of the comic book pages, and contains one or several speech bubbles or no speech bubbles at all. The images contain a total of 346 speech bubbles. All sample images included in this paper are copyrighted material of Fujiko-Fujio. They are used only to help readers understand the work described in this paper.

Each speech bubble is tagged manually by specifying its location in an image. Specifically, the location of a speech bubble is specified with the coordinates of the top left corner and the bottom left corner of a rectangle which surrounds the speech bubble. The detection of speech bubbles made by the systems are checked for correctness by comparing the coordinates of the bounding box of the speech bubbles humans specified, and the coordinates of the bounding box the systems specified.

When a speech bubble is detected, the expert system paints white pixels within the speech bubbles with a randomly picked color other than white. The machine learning system draws a rectangle around the speech bubble. Figure 5 shows an example image produced by the systems. Thus, the correctness of the detection of speech bubbles can also be visually checked.

Figure 5. Example image produced by the expert and machine learning systems

7.3. Evaluation of the expert system

The expert system is evaluated on the 243 images described above. Table 4 shows the result. The expert system segments the images and selects 804 regions as candidate regions. Of those regions, the expert system identifies 337 regions as speech bubbles correctly. It incorrectly identifies 32 regions as speech bubbles. Moreover, it identifies 8 regions as negative incorrectly. Overall, the precision of the expert system is 0.913, that is about 91% of the regions the expert system identified as speech bubbles are indeed speech bubbles. The recall is 0.977, which means about 98% of speech bubbles are detected. The F-score is 0.944.

Table 4. The performance of the expert system on the 243 images

	TP	FP	TN	FN	Accuracy	Precision	Recall	F-score
Expert	337	32	427	8	0.95	0.913	0.977	0.944

Figure 6. Examples of the images produced as the result of speech bubble detection

Figure 6 shows examples of the images produced as the result of the speech bubble detection. The first image from the left shows that all speech bubbles are correctly detected. The second image from the left is an example of the false negatives. The speech bubble in this image is missed because the edge of the bubble is broken and the

gap in the edge is more than 3 pixels wide. The third image from the left shows a case of the false positives. The lens of glasses are identified as speech bubbles because the eye contained in the lens has similar black pixel count and similar size to text. Adjusting the heuristic so that the glass would not be identified as a speech bubble would produce more false negatives. The fourth image from the left contains another example of the false positives. It is a false positive because the text is not contained in a speech bubble, which the system is supposed to detect. It is identified as a speech bubble because the text is completely surrounded by edges. Correcting this case would require detection of the shape of the edges surrounding the text.

Although the performance of the expert system is very good, we would like to have zero false negatives and 100% recall because a possible application of the expert system would be pre-processing step of a system which would process extracted text, where omission of any text would not be allowed.

7.4. Evaluation of the machine learning system

7.4.1. Supervised Learning

The same 243 images used in the evaluation of the expert system are used for evaluation of the machine learning system. The feature sets for the 804 regions the expert system have selected as candidate regions for speech bubbles are exported from the expert system. The 804 feature sets are divided in 2 to 1 ratio into 539 training sets, 266 test sets. Each of the three machine learning algorithms is trained on the training sets and evaluated on the test sets. The Maximum Entropy is trained multiple times (10 iterations and 20 iterations) before tested on the test sets. Table 5 shows the performance of the three algorithms averaged over 5 runs. Before each run of the

evaluation, the feature sets are randomly shuffled before divided into the training set and the test set.

All three algorithms produce results comparable to or better than that of the expert system. Especially, the Maximum Entropy and SVM do very well in terms of recall, both scoring above 0.98, which is better than the score of the expert system. Of the three algorithms the SVM produces the best result. The Naive Bayes does the worst of the three with scores lower than those of the expert system in all four metrics.

Table 5. The performance of the machine learning system in supervised learning

	Accuracy	Precision	Recall	F-score
Expert	0.95	0.913	0.977	0.944
Naive Bayes	0.908	0.897	0.889	0.892
Max Ent (10 iter.)	0.934	0.881	0.981	0.928
Max Ent (20 iter.)	0.928	0.863	0.985	0.920
SVM	0.943	0.900	0.973	0.935

When the results produced by the machine learning algorithms are visually checked, it is found that all the three algorithms tends to err in the cases in which the expert system errs. However, the Maximum Entropy and the SVM tends to produce more false positives than the expert system and thus, produce higher recall. They correctly detect some speech bubbles that are missed by the expert system.

The implementations of the Naive Bayes and Maximum Entropy algorithms included in the NLTK have a functionality to identify most informative features. Table 6 shows the 10 features the Naive Bayes classifier finds most informative. Table 7 shows the 10 features the Maximum Entropy classifier finds most informative.

Table 6. 10 most informative features for Naive Bayes classifier

	Name	Description
1	isRegionEnclosed	Whether or not the region is enclosed by black pixels == true OR The percentage the region is enclosed by black pixels > 83%
2	isXLessThanMax	The horizontal range of the region < 75% of the width of the image
3	isDistBtwCentersLessThanMax	The distance between the center of the region and the center of the group of black pixels the region contains < 57.4
4	isAreaMoreThanMin	1% < The percentage of the area of the image the region occupies
5	isPixCountMoreThanMin	The number of pixels > 6000
6	isYLessThanMax	The vertical range of the region < 75% of the height of the image
7	isAreaLessThanMax	The percentage of the area of the image the region occupies < 99%
8	histV_9	The 9th bin of the histogram of the count of black pixels binned into 10 vertically divided regions.
9	histV_8	The 8th bin of the histogram of the count of black pixels binned into 10 vertically divided regions.
10	pixCount	The count of pixels

Table 7. 10 most informative features for Maximum Entropy classifier

	Name	Description
1	percentArea	The percentage of the area of the image the region occupies
2	distBtwCenters	The distance between the center of the region and the center of the group of black pixels the region contains
3	isAreaLessThanMax	The percentage of the area of the image the region occupies < 99%
4	isRegionEnclosed	Whether or not the region is enclosed by black pixels == true OR The percentage the region is enclosed by black pixels > 83%
5	isPixCountMoreThanMin	The number of pixels > 6000
6	isXLessThanMax	The horizontal range of the region < 75% of the width of the image
7	isYLessThanMax	The vertical range of the region < 75% of the height of the image
8	isDistBtwCentersLessThanMax	The distance between the center of the region and the center of the group of black pixels the region contains < 57.4
9	isAreaMoreThanMin	1% < The percentage of the area of the image the region occupies
10	pixCount	The count of pixels

From the tables we can observe that the Naive Bayes finds binary features, which are inspired by the expert system, most informative while the Maximum Entropy finds real number value features as well as the binary features most informative.

7.4.2. Semi-Supervised Learning

In the semi-supervised learning, 403 feature sets extracted from 117 images are used. We try 1:1 and 2:1 training / test split.

The machine learning system is trained on the training sets and then tested on the test sets. Then the test sets with the labels assigned by the machine learning system are added to the training sets, and the machine learning system is trained on the new training sets. In the new training sets, the sets labeled by hand are given more weight than the sets labeled by the machine learning system itself. The weight is assigned by duplicating a set according to the weight. For example, a set whose weight is 2 is duplicated into two sets. The weight is assigned in this fashion in order to incorporate the weight into the Naive Bayes classifier, which does not use weights. We experiment with weights of 1 and 2.

The trained system is tested on 402 feature sets extracted from 123 images, which have not been seen by the machine learning system before. Table 8 shows the performance of the expert system on the same data. The accuracy of the expert system on the data is 0.935. The precision and the recall are 0.904 and 0.955 respectively. Each of the three machine learning algorithms which have been trained on the training data is tested on the test data. The Maximum Entropy is trained multiple times (10 iterations and 20 iterations) before tested on the test sets. Table 9 and Table 10 show the performance measures for the three machine learning algorithms used. The second column indicates the ratio of split between the training sets and the test sets and the weight assigned on the training sets when new training sets are produced by combining the training sets and

the test sets. For example, “1:1,2” means that the split between the training sets and the test sets is 1 : 1, and the weight assigned on the training sets is 2.

Table 8. The performance of the expert system on the same data used by the machine learning system

	TP	FP	TN	FN	Accuracy	Precision	Recall	F-score
Expert	170	18	205	8	0.935	0.904	0.955	0.929

Table 9. The performance of the machine learning system in semi-supervised learning – part 1

	Train.:Test, Weight on Train.	Accuracy	Precision	Recall	F-score
Naive Bayes	1:1,1	0.878	0.865	0.861	0.863
Naive Bayes	1:1,2	0.880	0.860	0.872	0.866
Naive Bayes	2:1,1	0.906	0.883	0.909	0.895
Naive Bayes	2:1,2	0.915	0.891	0.921	0.906
SVM	1:1,1	0.935	0.893	0.970	0.930
SVM	1:1,2	0.915	0.861	0.965	0.910
SVM	2:1,1	0.927	0.882	0.965	0.922
SVM	2:1,2	0.932	0.888	0.970	0.923

Table 10. The performance of the machine learning system in semi-supervised learning – part 2

	Train.:Test, Weight on Train.	Accuracy	Precision	Recall	F-score
Max Ent (10 iter.)	1:1,1	0.922	0.874	0.963	0.916
Max Ent (10 iter.)	1:1,2	0.915	0.859	0.967	0.910
Max Ent (10 iter.)	2:1,1	0.921	0.872	0.964	0.916
Max Ent (10 iter.)	2:1,2	0.920	0.869	0.967	0.915
Max Ent (20 iter.)	1:1,1	0.918	0.863	0.970	0.913
Max Ent (20 iter.)	1:1,2	0.916	0.865	0.961	0.911
Max Ent (20 iter.)	2:1,1	0.917	0.861	0.971	0.913
Max Ent (20 iter.)	2:1,2	0.922	0.873	0.965	0.917

All three algorithms produce results comparable to that of the expert system. Especially the Maximum Entropy and SVM do very well in terms of recall. The Maximum Entropy and the SVM do better than the expert system on recall. The F-score and the accuracy of the SVM are as good as those of the expert system.

The performance of the Naive Bayes improves as more correctly labeled training data is included in the training sets. However, it is very interesting to find that some machine learning algorithms (the Maximum Entropy and the SVM) do better with training sets

produced by assigning equal weight on the original training sets and the test sets. We suspect that it is because duplicating the same training data does not make any difference to the Maximum Entropy and the SVM. Specifically with the SVM adding the same data multiple times does not change the boundary between the positives and the negatives. Therefore, we suspect that the numbers differ in the four entries due to chance.

7.4.3. Ensemble Classifier

We also experiment with a ensemble machine learning method, in which multiple classifiers are combined to arrive at final classifications. We experiment with two variations of the ensemble method, in which classification results of two classifiers are combined.

In the first variation (Ensemble AND), if both of the classifiers agree that a sample is positive, the sample is classified as positive. Otherwise, the sample is classified as negative. In the second variation (Ensemble OR), if at least one of the classifiers classifies that a sample is positive, the sample is classified as positive. Otherwise, the sample is classified as negative.

The motivation for combining different classifiers is to get the best of each different classifier. From the classification results shown in Table 5, we know that the SVM is good at precision and the Maximum Entropy is good at recall. If we combine a classifier with good precision with a classifier with a good recall, we should be able to improve F-score.

We experiment with combining the SVM and the Naive Bayes classifiers, and the SVM and the Maximum Entropy classifiers. Table 11 shows the results, where Max Ent 10 iter. means the Maximum Entropy with 10 iterations and Max Ent 20 iter. means the Maximum Entropy with 20 iterations. All the classifiers are trained by the supervised learning method.

Table 11. The performance of the ensemble classifiers for speech bubble detection

	TP	FP	TN	FN	Accuracy	Precision	Recall	F-score
Expert	170	18	205	8	0.935	0.904	0.955	0.929
Ensemble AND (SVM + Naive Bayes)	166	14	209	12	0.935	0.922	0.932	0.927
Ensemble AND (SVM + Max Ent 10 iter.)	169	13	210	9	0.945	0.928	0.949	0.938
Ensemble AND (SVM + Max Ent 20 iter.)	170	13	210	8	0.947	0.928	0.955	0.941
Ensemble OR (SVM + Naive Bayes)	173	28	195	5	0.917	0.860	0.971	0.912
Ensemble OR (SVM + Max Ent 10 iter.)	173	33	190	5	0.905	0.839	0.971	0.900
Ensemble OR (SVM + Max Ent 20 iter.)	173	32	191	5	0.907	0.843	0.971	0.902

As shown in Table 11, we are able to achieve better results by combining the SVM and the Maximum Entropy classifiers. We get the best result by combining the classification results of the SVM and the Maximum Entropy with 20 iterations with logical AND operation, which performed better than the expert system. We get the best recall of 0.971 by combining the SVM and any of the other classifiers.

8. Eye Detection

Having successfully implemented our approach for the problem of speech-bubble detection, we experiment with implementing the same approach with another problem: character eye detection. Since eyes are usually drawn with round edges, we assume that an eye is a round white region surrounded by black edges, and that we can detect the eyes of the characters by finding pairs of the regions which are eyes. For each eye pair detected, we surround it with a rectangle box.

8.1. Image Processing

Regions surrounded by black edges are segmented using the aforementioned region growing algorithm. Every possible combination of the regions is formed and the features are extracted from each region and from the bounding box that bounds the pair of the regions. We call the pair of regions as member regions, and the region bounded by the bounding box, which encompass a pair of the member regions, as a candidate region. The horizontal coordinates of the top left and the bottom left corners of the candidate region are 3 pixels left of the most left coordinate of the member regions. The horizontal coordinates of the top right and the bottom right corners of the candidate region are 3 pixels right of the most right coordinate of the member regions. The vertical coordinates of the top left and the top right corners of the candidate region are 3 pixels above the highest coordinate of the member regions. The vertical coordinates of the bottom right and the bottom left corners of the candidate region are 3 pixels below the lowest coordinate of the member regions. This is done in order to include the black edges that surround the member regions into the candidate region because the region growing stops 2 pixels before the black edges and therefore, the member regions do not include the black edges that enclose the member regions.

8.2. Features

Table 12, 13, and 14 show the features used for the eye detection.

Table 12. Features used in the eye detection – part 1

#	Feature Name	Description
1	diffArea	Difference of the areas between the pair of regions
2	diffX1	Difference of the left coordinates between the pair of regions
3	diffX2	Difference of the right coordinates between the pair of regions
4	diffY1	Difference of the top coordinates between the pair of regions
5	diffY2	Difference of the bottom coordinates between the pair of regions
6	distCenters	The distance between the centers of the pair of regions
7	distCorners	The distance between the corners of the pair of regions
8	distX	The horizontal distance between the pair of regions
9	distY	The vertical distance between the pair of regions
10	isInsideRegion	Whether or not any of the regions are inside other segmented regions
11	diffPercentArea	The difference between the percentages of area each region of the pair occupies in the image
12	avPercentArea	The average of the percentages of area each region of the pair occupies in the image
13	diffBpRatio	The difference between the ratio of black pixels over all the pixels in each region of the pair
14	avBpRatio	The average of the ratio of black pixels over all the pixels in each region of the pair
15	top left	Whether or not the top left corner of the bounding box touches the region
16	top right	Whether or not the top right corner of the bounding box touches the region
17	bottom left	Whether or not the bottom left corner of the bounding box touches the region

Table 13. Features used in the eye detection – part 2

#	Feature Name	Description
18	bottom right	Whether or not the bottom right corner of the bounding box touches the region
19	left	Whether or not the middle of the left edge of the bounding box touches the region
20	right	Whether or not the middle of the right edge of the bounding box touches the region
21	top	Whether or not the middle of the top edge of the bounding box touches the region
22	bottom	Whether or not the middle of the bottom edge of the bounding box touches the region
23	x overlap	Whether or not the bounding box overlaps horizontally
24	y overlap	Whether or not the bounding box overlaps vertically
25	sumHistH	The sum of the HistHs (The histogram of the count of black pixels binned according to their horizontal positions) from the pair of regions
26	sumHistV	The sum of the HistVs (The histogram of the count of black pixels binned according to their vertical positions) from the pair of regions
27	sumHistR	The sum of the HistRs (The histogram of the count of black pixels surrounding the group of the black pixels marked as text binned according to their radial positions) from the pair of regions
28	diffHistH	Whether or not the both regions have black pixels in the same bin of the HistH (2 if the both have black pixels, 1 if one of them has the pixels, 0 if none have the pixels)
29	diffHistV	Whether or not the both regions have black pixels in the same bin of the HistV (2 if the both have black pixels, 1 if one of them has the pixels, 0 if none have the pixels)
30	diffHistR	Whether or not the both regions have black pixels in the same bin of the HistR (2 if the both have black pixels, 1 if one of them has the pixels, 0 if none have the pixels)
31	edgeMap	The histogram of the count of black pixels binned into 10 x 10 cells

Table 14. Features used in the eye detection – part 3

#	Feature Name	Description
32	atan	The histogram of atan2 of the edges in the regions of the pair binned into 4 bins of edge direction: horizontal, top right - bottom left, vertical, top left - bottom right
33	atanHistH	The histogram of atan2 of the edges in the regions of the pair binned according to their horizontal positions into 4 bins of edge direction: horizontal, top right - bottom left, vertical, top left - bottom right
34	atanHistV	The histogram of atan2 of the edges in the regions of the pair binned according to their vertical positions into 4 bins of edge direction: horizontal, top right - bottom left, vertical, top left - bottom right

The features #1 - 9 and #11 - 14 are to capture the difference between the properties of a pair of member regions. For example, the feature #1 (diffArea) is the difference between the area of the two member regions. The feature #2 (diffX1) is the difference between the left most coordinates, or the left coordinates of the bounding boxes of the member regions.

The feature #10 (isInsideRegion) describes whether or not any of a pair of member regions is contained within other regions. If two of them are inside other regions, the value of the feature is 2. If one of the member regions is inside other regions, the value is 1. If none of the member regions are inside other regions, the value is 0.

The features #15 - 22 are to capture the shape of the member regions. They describe whether or not four corners and the middle section of the four edges of the bounding boxes of the member regions are on the member regions or not. For example, if the shape of a member region is round, the four corners of the bounding box of the member region should not touch the member region, but the middle section of the four edges of

the box can touch the member region, as shown in Figure 7. For example, if the top left corners touch the member regions for the both regions, the value of the feature #15 is 2.

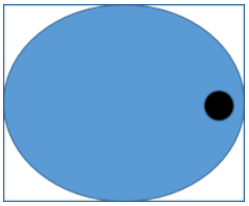

Figure 7. A segmented white region and its bounding box

The features #23 and 24 describe whether or not the two member regions overlap horizontally or vertically.

The features #25 - 34 are histograms. The features #25 - 27 are histograms of the sum of the count of black pixels in the two member regions. The histograms of the features #28 - 29 capture the differences between the histograms of black pixel count of the two member regions. For example, if the both corresponding bins have values greater than 0, it is represented with the value 2.

The feature #31 is a histogram of the count of black pixels in the candidate regions, which encompasses the member regions. The area of the box is divided into 10 x 10 cells and the count of black pixels in each cell is binned into the corresponding bin in the histogram.

The features #32 - 34 are histograms of atan2 of the edges in the candidate region. The atan2 of the edges are binned into 4 bins according to their directions: horizontal, top right - bottom left, vertical, top left - bottom right. The atan2 is calculated using java.lang.Math.atan2 function. The function takes two arguments: gradients Gx and Gy. The gradient Gx is computed by convoluting the pixels values of the region with the

sobel operator (Gonzalez & Woods, 2002, p. 578) shown in Figure 8 (a). The gradient Gy is computed by convoluting the pixel values of the candidate region with the sobel operator shown in Figure 8 (b). For example if the result of the atan2 of an edge in the candidate region is 45 degree, the value of the bin for 45 degree is incremented by 1.

(a)

-1	0	1
-2	0	2
-1	0	1

(b)

1	2	1
0	0	0
-1	-2	-1

Figure 8. Sobel operators

8.3. Algorithms

The aforementioned SVM and Naive Bayes algorithms are used to detect eyes. In addition to the two machine learning algorithms, we experiment with the AdaBoost boosting algorithm (Freund & Schapire, 1995).

8.3.1. Experimentation with AdaBoost

The AdaBoost combines multiple classifiers and boost the performance of the classifiers by dynamically adjusting the weights of training samples based on the classification errors. The classifiers used in the AdaBoost are called weak learners (Bishop, 2006, pp. 657-663) because they are machine learning algorithms that perform slightly better than chance. In our implementation, we use decision stumps (Bishop, 2006, p. 659) as the

weak learners. A decision stump is a simple classifier which classifies samples based on which side of a threshold they fall on.

We implement two AdaBoost variations. The first version of our AdaBoost implementation (we call it AdaBoost1) trains one decision stump on each feature and pick the best decision stump with the lowest error rate. The training process is iterated the number of times specified manually. The weights of the training samples are adjusted based on the errors made by the best decision stump after each iteration. The weights of the training samples that the decision stump classified incorrectly are increased and those of correctly classified samples are decreased. Thus, the decision stumps pay more attention to the samples that are misclassified in the previous iteration. After the training, the best decision stumps are combined and the final classification is decided by the combined vote of the best decision stumps whose voting weights are decided by their error rates.

Our second implementation of AdaBoost (we call it AdaBoost2) trains a series of decision stumps in sequence, one for each feature. The best decision stump with the smallest error rate is picked for each feature. After one decision stump is trained on one feature, the weights of the training samples that the decision stump classified incorrectly are increased and those of correctly classified samples are decreased, and then the next decision stump is trained on the next feature. Thus, the next decision stump pays more attention to the samples that the previous decision stump misclassified. We set the initial weights of the training samples 5 times larger (5.0 from 1.0) than those of the AdaBoost1 because we have discovered empirically that it enables us to get high recall with low false positives. We experiment with even higher starting weights, but we find that 5.0

works the best. The number of the possible order of features on which the classifiers are trained is N!, where N is the number of features. Since it takes very long time to try all the possible orders, we pick certain number (40 in our experiment) of the orders randomly and pick the one that produces the lowest error rate. After the decision stumps are trained, test data was classified by combining the decisions of the decision stumps. Voting weight is assigned to each decision stump based on its error rate: the decision stumps with low error rate are given larger voting weights than the decision stumps with higher error rates. The class label is decided by the weighted votes of the decision stumps.

8.4. Evaluation

We experiment with both the supervised learning and the semi-supervised learning. As in the case of the speech bubble detection, semi-supervised classifiers are trained on data labeled by supervised classifiers.

8.4.1. Supervised Learning

Table 15 shows the result for the supervised learning. All classifiers achieve high accuracy. However, among all possible pairs of the regions, less than 5% of them are actually eyes. Therefore, majority of the samples are negatives, and the classifiers can achieve accuracy of around 95% by simply classifying all the samples as negatives. Thus, metrics other than accuracy need to be looked at to evaluate the performance of the classifiers properly.

Other metrics are not as good as accuracy. The F-Score for all three classifiers are below 0.61. The SVM achieves the highest F-Score of 0.604 and the second highest precision of 0.629, but its recall is the second lowest with 0.583. On the other hand the AdaBoost2 classifier achieves the second highest F-score of 0.579 and the second highest Recall of 0.666, and the precision is 0.532. In the training of the AdaBoost2, 40 orders of features are picked randomly and the best one is picked. The Naive Bayes classifier produces the second lowest F-Score, but its recall is the highest with 0.672. The AdaBoost1 produces the lowest F-Score of 0.281 because its recall is very low.

Table 15. Eye detection - supervised learning

	Accuracy	Precision	Recall	F-score
SVM	0.960	0.629	0.583	0.604
Naive Bayes	0.930	0.413	0.672	0.510
AdaBoost1	0.951	1.000	0.166	0.281
AdaBoost2	0.948	0.532	0.666	0.579

Figure 9 shows an example of eyes detected by the classifiers. The classifiers are relatively good at detecting eyes which are round and adjacent to each other horizontally, however they tend to classify any round regions as eyes, as the right hand and the right eye of the left character are marked as a pair of eyes in the figure.

Figure 9. An example of eyes detected by the classifiers

8.4.2. Semi-Supervised Learning

Table 16 shows the result for the semi-supervised learning. In semi-supervised learning, the Naive Bayes classifier achieves the second highest F-Score of 0.57 with 71.3% recall and 47.5% precision. The AdaBoost2 achieves the highest F-Score of 0.582 with 57.6% precision and 63.1% recall. The AdaBoost1 produces a very good recall, but its precision is very low and as a result its F-Score is the lowest of the four classifiers. The SVM achieves the second lowest F-Score of 0.554 with 60.7% precision and 50.9% recall.

Table 16. Eye detection - semi-supervised learning

	Accuracy	Precision	Recall	F-score
SVM	0.956	0.607	0.509	0.554
Naive Bayes	0.938	0.475	0.713	0.570
AdaBoost1	0.744	0.150	0.738	0.250
AdaBoost2	0.947	0.576	0.631	0.582

8.4.3. Ensemble Classifier

We also experiment with the ensemble machine learning method, in which multiple classifiers are combined to arrive at final classifications. We experiment with two variations of the ensemble method, in which classification results of two classifiers are combined.

In the first variation (Ensemble AND), if both of the classifiers agree that a sample is positive, the sample is classified as positive. Otherwise, the sample is classified as negative. In the second variation (Ensemble OR), if at least one of the classifiers classifies that a sample is positive, the sample is classified as positive. Otherwise, the sample is classified as negative.

We experiment with combining the SVM and the Naive Bayes classifiers, and the SVM and the AdaBoost2 classifiers. Table 17 and Table 18 show the results. All the classifiers are trained by the supervised learning method.

Table 17. The performance of the ensemble classifiers for eye detection – part 1

	TP	FP	TN	FN	Accuracy	Precision	Recall	F-score
Ensemble AND (SVM + Naive Bayes)	57	0	1397	28	0.981	1.000	0.670	0.802
Ensemble AND (SVM + AdaBoost2)	50	0	1397	35	0.976	1.000	0.588	0.741

Table 18. The performance of the ensemble classifiers for eye detection – part 2

	TP	FP	TN	FN	Accuracy	Precision	Recall	F-score
Ensemble OR (SVM + Naive Bayes)	85	63	1334	0	0.957	0.574	1.000	0.729
Ensemble OR (SVM + AdaBoost2)	85	29	1368	0	0.980	0.746	1.000	0.855

The Ensemble AND achieves 100% precision. The AND combination of the SVM and the Naive Bayes produces the F-Score of 0.802, which is higher than that of the SVM and the AdaBoost2, because its recall is higher than that of the SVM and the AdaBoost2.

The Ensemble OR achieves 100% recall. The OR combination of the SVM and the AdaBoost2 produces the highest F-Score of 0.855 with 74.6% precision and 100% recall. The OR combination of the SVM and the Naive Bayes produces the F-Score of 0.729 with 57.4% precision and 100% recall.

Thus we are able to improve the performance of our eye detection system by combining multiple classifiers. We are able to improve the F-Score from 0.604, which is the best score obtained with the SVM to 0.855 by combining the SVM and the AdaBoost2 classifiers.

8.4.4. Informative Features

Figure 10, 11, and 12 show the plots of the histogram features #25 - 31.

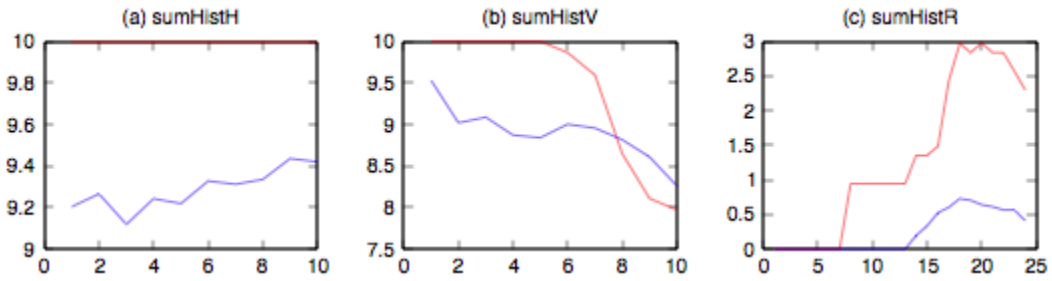

Figure 10. Plot of histograms (H for horizontal, V for vertical and R for radial) averaged over all samples for positives (red) and negatives (blue) - part 1

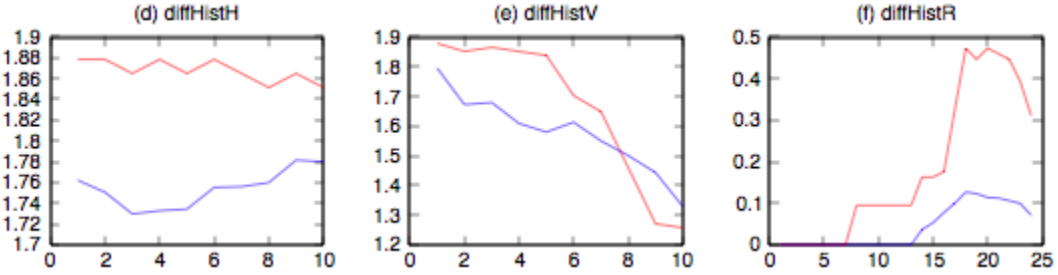

Figure 11. Plot of histograms (H for horizontal, V for vertical and R for radial) averaged over all samples for positives (red) and negatives (blue) - part 2

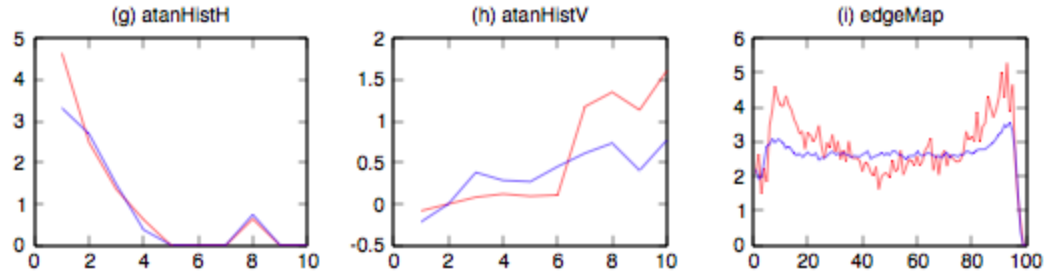

Figure 12. Plot of histograms (H for horizontal, V for vertical and R for radial) averaged over all samples for positives (red) and negatives (blue) - part 3

The values in the histograms are averaged over all the samples for positive samples and negative samples, respectively. The red lines show values of the positive samples and blue lines show values of the negative samples. The plots for the features #25 (sumHistH), #26 (sumHistV), #27 (sumHistR), #28 (diffHistH), #29 (diffHistV), and #30 (diffHistR) show that those histograms can be good features to separate positives from negatives. For example, the diffHistR and the atanHistH are ranked within the 10 most informative features by the Naive Bayes classifier. However, some of the histogram features are not found very informative. It looks as though we could separate positives from negatives solely by the values of the sumHistH, but the values for many negative samples are 10 as well. Therefore, we need to find better features in order to improve the detection rate.

9. Conclusion

In this thesis we present an expert system that detects the elements very accurately and a machine-learning system that is trained on features used by the expert system and does the same job as well as the expert system. For very difficult tasks such as Computer Vision problems presented in this paper, there is a limit with what an expert system can do. We can come up with a heuristic that works for certain set of data, but we soon realize that the heuristic does not work for all cases and it needs to be changed when we add new example. With machine learning, we no longer need to change our program to accommodate new examples; we let the machine learn from the examples by itself. We have shown in this paper that machine learning can be successfully applied to a very difficult task such as detecting elements present in comic book panels, and they can do the task as well as the expert system. We also apply the same approach to eye detection of characters in the panels. We use the Naive Bayes, SVM, and AdaBoost algorithms for the task. We are able to detect majority of the eyes but with low precision. The features used to train the machine learning classifiers are not good enough to separate positive samples from negative samples. Therefore, features that can separate positive samples from negative samples better need to be found in order to improve the eye detection rate. We also experiment with combining multiple classifiers. We are able to improve the performance of our eye detection system significantly by combining the SVM and either the Naive Bayes or the AdaBoost classifiers.

10. Future Work

We are able to achieve a significant rate for detecting eyes, but it is probably not good enough to be useful in some real world applications. In order to improve the detection rate further, robust features need to be discovered. Future work will strive toward discovering robust and efficient image segmentation methods which can segment not only eyes but also faces or characters as a whole.

The focus of this thesis was on speech bubbles and eyes. For automatic translation of text, the detection of the speech bubbles would be sufficient. However, for the automatic narration of stories depicted in comic books and the search of characters depicted in images, each character needs to be recognized and the speech bubbles need to be associated to the characters who made the speech. Thus, the next step would be to expand the systems presented here to detect not only eyes but also whole characters present in the images. After the characters are detected, the next step would be to recognize characters and to associate speech bubbles with characters.

An interesting future goal is to be able to detect even more elements in each panel. If the machine could detect all the elements present in the panels, the scenes depicted in the panels could be described automatically. For example, if the machine could detect that a wizard holding a wand is depicted in a panel, it could automatically describe the scene as “A wizard is holding a wand.” Hence, another step in the future would be to expand the system to detect other elements in the panels.

BIBLIOGRAPHY

1. Bird, S., Klein, E., & Loper, E. (2009). Natural language processing with Python. Sebastopol: O'Reilly.
2. Bishop, C. (2006). *Pattern recognition and machine learning*. New York: Springer.
3. Bowker. (2014). *Print isn't dead, says Bowker's Annual Book Production Report*. Retrieved June 9, 2014 from http://www.bowker.com/en-US/aboutus/press_room/2011/pr_05182011.shtml
4. Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-8*(6), 679-698.
5. Eitz, M., Hays, J., & Alexa, M. (2012). How Do Humans Sketch Objects? *ACM Transactions on Graphics (Proc. SIGGRAPH), 31*(4), 44:1-44:10.
6. Fisher, J., Hinds, S., & D'Amato, D. (1990). *Proceedings 10th International Conference on Pattern Recognition: 16-21 June 1990, Atlantic City, New Jersey, USA.* (Vol. 1, pp. 567 - 572). Los Alamitos, Calif.: IEEE Computer Society Press.
7. Fletcher, L., & Kasturi, R. (1988). A robust algorithm for text string separation from mixed text/graphics images. *IEEE Transactions on Pattern Analysis and Machine Intelligence,10*(6), 910-918.
8. Freund, Y. & Schapire, R. E. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. *Computational Learning Theory: Eurocolt 95*, Springer-Verlag, 23-27.
9. Gonzalez, R., & Woods, R. (2002). *Digital image processing* (2nd ed.). New Jersey: Prentice Hall.
10. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. & Witten, I. H. (2009). The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11(1).

11. Horn, B. (1986). *Robot Vision* (MIT Press ed.). Cambridge, Mass.: MIT Press.
12. Hu, C., Quinn, A. J., Rose, A., Bederson, B. B. & Arisaka, T. (2008). Enhancing Readability of Scanned Picture Books. *Human-Computer Interaction Lab - 2008-09*. http://www.cs.umd.edu/localphp/hcil/tech-reports-search.php?number=2008-09
13. Liu, B. (2011). *Web data mining exploring hyperlinks, contents, and usage data* (2nd ed.). Berlin: Springer.
14. NLTK Project. (2014). *NLTK 3.0 Documentation*. Retrieved August 21, 2014 from http://www.nltk.org/
15. Parr, B. (August 5, 2010). *Google: There Are 129,864,880 Books in the Entire World*. Retrieved from Mashable website: http://mashable.com/2010/08/05/number-of-books-in-the-world/
16. Peng, H., Long, F. & Chi, Z. (2003). Document Image Recognition Based on Template Matching of Component Block Projections. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 25(9), 1188-1192.
17. Peng, H., Chi, Z., Siu, W. & Feng, D. D. (2000). PageX: An Integrated Document Processing and Management Software for Digital Libraries. *Proc. 2000 Int'l Workshop Multimedia Data Storage, Retrieval, Integration, and Applications*, 203-207.
18. Tombre, K., Tabbone, S., Pélissier, L., Lamiroy, B. & Dosch, P. (2002). Text/Graphics Separation Revisited. *DAS '02 Proceedings of the 5th International Workshop on Document Analysis Systems V*, 200-211.
19. UNESCO. (n.d.). *Index Translationum*. Retrieved June 9, 2014, from http://www.unesco.org/xtrans/bsstatexp.aspx?crit1L=4&nTyp=min&topN=50
20. Viola, P. & Jones, M. (2004). Robust Real-Time Face Detection. *International Journal of Computer Vision*, 57(2), 137-154.

21. Wong, K. Y., Casey, R. G. & Wahl, F. M. (1982). Document Analysis System. *IBM J. Res. Develop.*, 2(6), 647-656.