Identifying Diet-Restricted Ingredients in Food Labels Using A Pattern Matching OCR Algorithm

Elliot Catalano

Abstract

Many individuals maintain restricted dietary restrictions for allergenic, health-related, or ethical reasons. To adhere to a strict diets, it is often necessary to closely read ingredient labels to monitor for any potentially threatening ingredient. This can cause a major amount of inconvenience, especially for someone with an extensive list of ingredients to monitor. Instead of users meticulously scouring labels, a user may instead be aided with the help of visual image processing. Optical character recognition (OCR) offers a solution to accurately detect and recognize text within food labels. In this experiment, an OCR algorithm that utilized pattern matching was trained and used to detect and recognize text within several sample ingredient labels. Recognized text was then compared to lists of allergy and diet restricted ingredients to detect any unsafe products. When properly trained and under ideal circumstances, the algorithm that was tested worked nearly flawlessly. Under practical conditions, however, this algorithm proves to be insufficient for pragmatic use. Many improvements need to be made in order to utilize a pattern-matching OCR algorithm to reliably identify ingredients within a food label.

1. Introduction

Image processing is the process of performing operations on a digital image in order to extract useful data and information. The area of image processing is a large and growing field with countless uses, ranging from simple entertainment to vital application in security and medicine. Image processing affects our daily lives. Some examples of facial recognition being used are: in Facebook for automatically tagging people within photos, in Snapchat for placing filters on faces, in online classes to take attendance, and in hospitals to help diagnose certain medical conditions. As we continue to research and improve on algorithms and techniques in image processing, we find more and more ways to use image processing to improve our lives.

Food allergies and dietary restrictions are having more of an emphasis on the lives of average people. One extremely important part of maintaining a healthy and well-tuned body and

mind is to utilize a well-rounded and healthy diet. With the differences in ethical and health-related areas between people, a specific "healthy" diet is hard to identify on large scale. To suit individual needs, the fine tuning of a diet may be best to create and maintain an optimal diet regiment. Two large sources of major variations between diets are: ethical reasons and allergenic reasons.

An allergy is an immunological response of the body cause by the presence of a specific foreign substance. In most cases, immunological responses are beneficial for protecting the body. In the case of allergies, the responses are caused by "false alarms," and may cause effects that are extremely uncomfortable or life threatening. Some of the most common substances that cause allergic reactions are: pollen, pet dander, mold spores, insect stings, food, and medicine (MedlinePlus). For someone who is allergic to a specific set of food ingredients, it is often necessary to closely monitor food labels and ingredient lists to prevent the consumption of a potentially dangerous substance. According to a study conducted in 2009 by the American Academy of Allergy, Asthma, and Immunity (AAAAI), 8% of people have a food allergy. Of this 8% of people with food allergies, 38.7% have a history of sever reactions. It was also found that 30.4% of food allergic children have multiple food allergies (AAAAI). Also according to AAAAI, the prevalence of allergies worldwide has increased for over 50 years. The issue of food allergies is an extremely prevalent and growing issue within the general population.

Ethical differences between people in relation to food sources is a common source of debate within different groups following different diets. In the United States, the population of people following a vegan diet is rapidly growing. Nearly 6% of Americans identify as vegans, which is about a 500% increase since 2014 (VeganBits). According to a report titled, *Top Trends in Prepared Foods in 2017*, two major trends were meat-free eating and ethical eating, meat-free

being the #1 trend overall (GlobalData). As the split continues to grow within the population, it becomes more necessary for people to closely monitor ingredient labels for diet-restricted ingredients. According to the PETA list of animal-derived ingredients, there are over 250 ingredients that vegans must look out for in order to completely rid themselves of animal-based food sources.

Visual image processing may be used in order to alleviate the troubles of maintaining a specific diet and avoiding consuming any undesired ingredients. Text recognition, for example, can be used as a tool to help avoid endlessly reading ingredient labels to search for diet-friendly products. Optical character recognition, for example, provides a method for processing an image and extracting text. In this paper, I explore an OCR algorithm and its effectiveness of reading and extracting ingredients from the ingredient lists of food labels.

2. Methodology

In order to implement optical character recognition to detect and identify text within an image, my algorithm was based on an OCR algorithm explored in the paper "Optical Character Recognition Implementation Using Pattern Matching" published in the International Journal of Computer Science and Information Technologies in 2014. The basic algorithm described in the paper was comprised of 4 distinct steps: greyscaling, feature extraction, recognition of pattern, and recognition of output. In my implementation, I added several steps: resulting in: preprocessing, feature extraction, recognition of pattern, recognition of output, parsing of output, and comparing the parsed output with a preset list of non-vegan ingredients.

2.1 Preprocessing

First, an image was turned to greyscale from RGB using the formula: $Y = 0.2126R+0.7152G+0.0722B$, where Y represents the greyscale value. Next, the image was binarized using image thresholding, making text pixels a greyscale value of 0, and any other pixels a greyscale value of 255. Thresholding was done by taking a histogram of all greyscale color information, finding the value with a highest histogram value. The value with the highest histogram value, along with any histogram value +45 and -65 was determined as a background pixel and turned white. Any remaining pixel was then determined as a text pixel and turned black. Any remaining "noise" of stray black pixels was removed by iterating through the pixel, searching for stray black pixels, and turning them white. Stray pixels were found by iterating through the image and searching for pixels of value 0 with no neighbors to the left, right, top and bottom of value 0. Figure 1 shows an example of a stray pixel vs. a non-stray pixel.

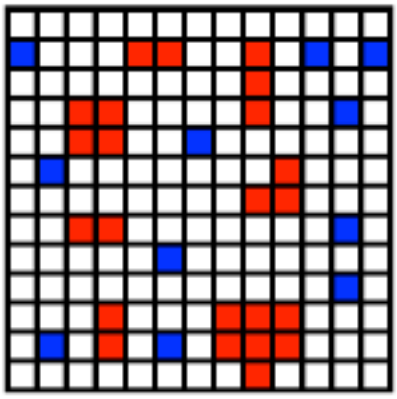

Figure 1. Stray vs. Non-Stray Pixel
Stray pixels are denoted by blue, non-stray pixels by red.

2.2 Feature Extraction

To extract the features of an image, the preprocessed image was first segmented into horizontally oriented sections denoting lines of text. This was done by iterating through an image horizontally. If a pixel of value 0 was found, then the top coordinate of a line was recorded. Next,

I continued iterating through an image until a row of completely white pixels was found, denoting the bottom coordinate of a line. The tops of bottoms of each line was stored in arrays as found. Figure 2 illustrates the top and bottom coordinates of a line.

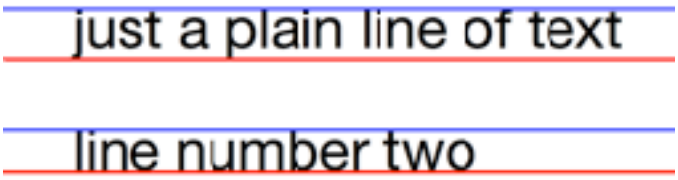

Figure 2. Top and Bottom Coordinates of a Line
Note: The coordinate maybe lower than the bottom of each individual letter within the line

Next, the leftmost pixel of each letter was found by iterating vertically through each line until a black pixel was found, denoting the left coordinate of a character. The rightmost pixel of a letter was then found by continuing iteration until a column of white pixels was found, marking the right coordinate of a character. The left and right coordinates were then stored in an array for each line. The top, bottom, left, and right coordinate arrays could then be used to find the general outlines of each letter. Figure 3 illustrates the left and right coordinates for each letter in a word.

Figure 3. Top, Bottom, Left, and Right Coordinates of Each Letter

Because the top and bottom coordinates of each letter were uniform for the entire line, many letters could have whitespace above and below them. This is due to other letters being taller or falling lower than it in the line. To adjust from it, I first cropped the letter using the previously

obtained top, bottom, left, and right coordinates. Next, I iterated through the cropped image horizontally starting from the top until the first black pixel is found. This marked the location for an adjusted top coordinate. Next, I iterated through the cropped image horizontally starting from the bottom, until the first black pixel is found. This marked the location for an adjusted bottom coordinate. Using the adjusted top and bottom coordinates, along with the previously found left and right coordinates, we can obtain an accurate cropping of each individual letter. Figure 4 shows an example of a cropped letter.

Figure 4. Cropped Letter Using Adjusted Top, Adjusted Bottom, Left, and Right Coordinates
Note: The grey curved corner is a result of the curved corners on any window on macOS Sierra

After finding the cropped letter, each letter was resized into a 15x15 image using OpenCV's resize function.

2.3 Recognition of Pattern

After resizing the letter, a 2d matrix was used to story binary information about the image. This was done by iterating through the image, and every white pixel was assigned value 1 in the 2d matrix, while every black pixel was assigned value 0 in the 2d matrix. Figure 5 shows the corresponding 2d matrix for the letter "S".

Figure 5. Binary Matrix For a Letter

After creating a binary matrix of a letter, the distance of the farthest pixel from the center was found using the formula [D = sqrt((y2 – y1)^2 + (x2 – y1)^2)]. This distance was then used to calculate the imaginary track using the formula: T = D/5. Next, 8 imaginary sectors were created by separating a 15x15 matrix into 8 sections as shown in figure 6. To increase the amount of information obtained by the sectors, I also divided the separating lines into 8 sections, as shown in figure 7. This allowed for 49 extra spaces, or 21.8% more information being collected.

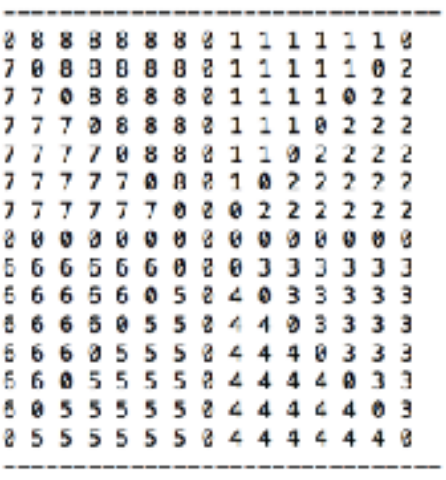

Figure 6. Original Imaginary Sectors

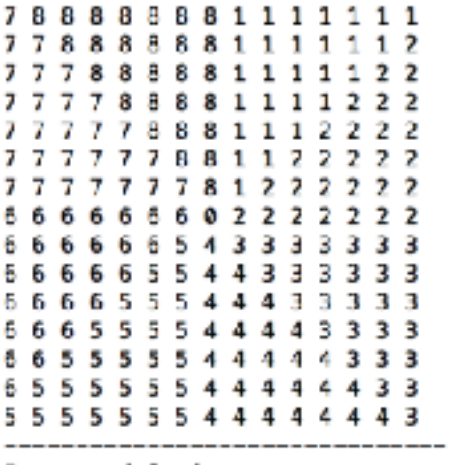

Figure 7. Adjusted Imaginary Sectors

By collecting information about the number of 0's in the intersection of a letter's sectors and tracks, templates about letters can then be created and used to identify text in images. To train a set of data, templates are created for all letters and special characters. The track-sector information about each letter is then stored in a separate text file.

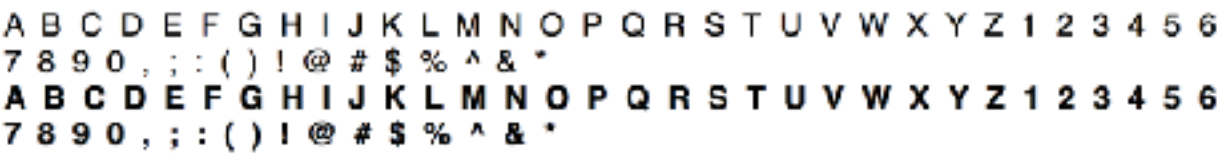

Figure 8. Training set for Alphabet and Special Characters.

2.4 Recognition of Output

After a templates is made for each letter and special characters by using a training set, the sector-track information of each individual letter within an image is compared to the information stored within a text file and used to identify each letter. By iterating through each letter in an image, identifying the output, and appending the output to a string, I generated an extracted string for all the text in the image. Figure 9 shows an example of the extracted string for an image.

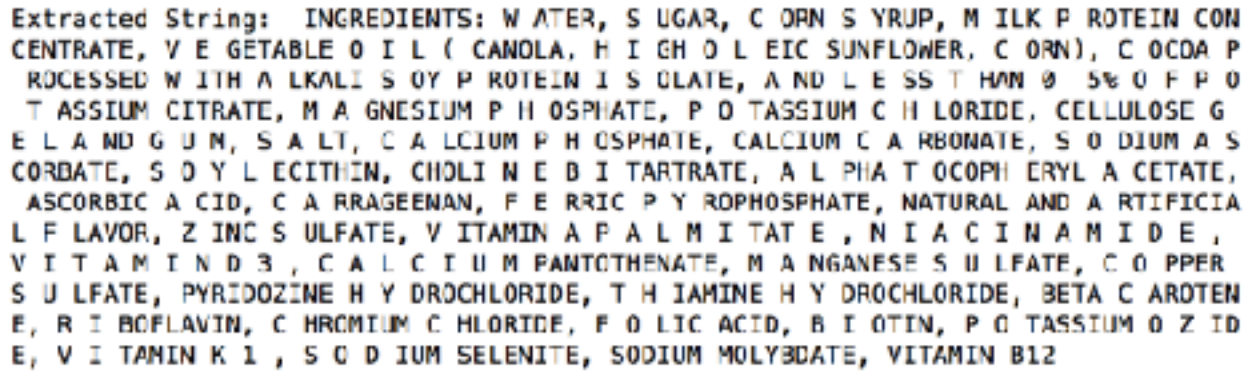

Figure 9. Extracted String of an Image

2.5 Parsing of Output

After obtaining an extracted string for all text in an image, the string was parsed by separating ingredients by the presence of a comma (“,”) and stored in a vector. At the end of parsing the extracted string, the remaining vector was iterated through and analyzed to find any non-vegan ingredients it may contain.

2.6 Identification of Non-Vegan Ingredients

In order to determine if each ingredient was non-vegan, first a list of animal based ingredients was collected from PETA’s animal derived ingredients list (**PETA**). This list, however, was not completely exhaustive, and was merely used as an example for analysis. An

array of strings containing each ingredient on PETA's list was created and used to compare to all ingredients in the extracted vector from the ingredients picture. To compare strings, first I converted all extracted strings and all non-vegan ingredient strings into lowercase, in order to avoid any complications due to capitalization. Next, I iterated through the list of non-vegan ingredients and determined if any were a substring of the extracted string. If it was a substring, of the extracted string, the non-vegan ingredient was returned and printed out.

3. Results

In tests, this algorithm proved to be unreliable for use in the average case. One issue that I found immediately when testing sets of data was the specificity of training data. In order to accurately detect letters within an image, the training data must be from the same font, with the same character styles (italic, bold, underline, etc.) and same font size. Secondly, this algorithm requires that letters do not utilize kerning. Kerning is the overlapping of certain portions within letters. Figure 10 illustrates kerning within letters. Kerning is problematic because the algorithm used searches for a vertical like of all white pixels to separate letters. As shown in figure 11, this algorithm sometimes recognizes multiple letters with kerning as single letters.

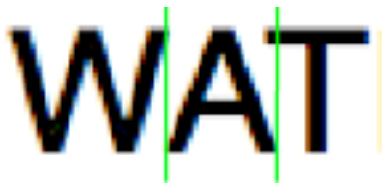

Figure 10. Letters With Kerning

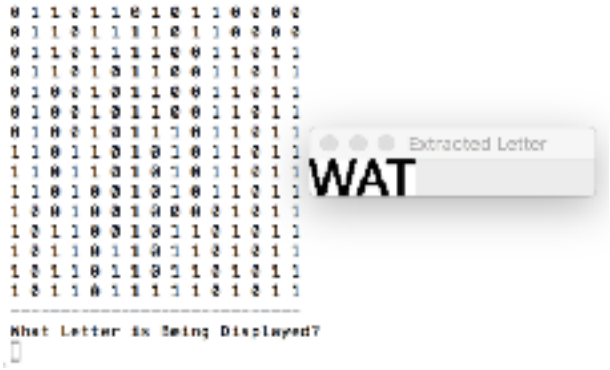

Figure 11. Problematic Kerning

A similar issue that arose was the presence of connected letters on ingredients lists due to inaccurate printing. Figure 12 illustrates a real example of this on an ingredients list.

Figure 12. Connected Letters on Real Food Label

In order to prevent this issue in test samples, I had to increase the character spacing between letters to 14%, while the standard is 0%. One last issue that arose were false positives being raised. One example was the substring “fat” being found in multiple ingredients, such as “Zinc Sulfate”. This can be easily fixed by creating a list of ingredients that must exactly match an extracted string.

4. Discussion / Conclusion

Overall, this algorithm worked nearly 100% of the time when trained on test data that matched the input image and when letters within the input image was properly spaced out. For practical uses, however, a pattern matching algorithm must be properly trained with many fonts and sizes in order to accurately identify letters within an input image. For accurate identification of non-diet friendly ingredients, an exhaustive list of restricted ingredients must be created. For further implementation and research, it would be best to attempt to utilize machine learning and neural networks to categorize letters and words.

References

AAAAI. Allergy Statistics. Retrieved August 13, 2017, from http://www.aaaai.org/about-aaaai/newsroom/allergy-statistics

Anbarjafari, G. (n.d.). Introduction to image processing. Retrieved August 13, 2017, from https://sisu.ut.ee/imageprocessing/book/1

PETA. Animal-Derived Ingredients Resource. Retrieved August 13, 2017, from https://www.peta.org/living/other/animal-ingredients-list/

Center for Food Safety and Applied Nutrition. (n.d.). Labeling & Nutrition - 7. Nutrition Labeling; Questions L1 through L153.

GlobalData. (n.d.). Top Trends in Prepared Foods 2017: Exploring trends in meat, fish and seafood; pasta, noodles and rice; prepared meals; savory deli food; soup; and meat substitutes.

MedlinePlus. Allergies. Retrieved August 13, 2017, from https://medlineplus.gov/allergy.html

Mohammad, F., Anarase, J., Shingote, M., & Ghanwat, P. (2014). Optical Character Recognition Implementation Using Pattern Matching. International Journal of Computer Science and Information Technologies, 5, 2088-2090.

VeganBits. (2017, May 23). Vegan Demographics 2017 - USA, and the world. Retrieved August 13, 2017, from http://veganbits.com/vegan-demographics-2017/

Vynckier, I. (n.d.). Segmenting Words and Characters | How OCR Works. Retrieved August 13, 2017, from http://www.how-ocr-works.com/OCR/word-character-segmentation.html