A Survey of Recursive Learning Approach for License Plate Matching with Normalized Edit Distance Metric

Anand Chourey[1], Avinash Sharma[2], Madhuvan Dixit[3]
[1]M.Tech Student, [2]Associate Professor (HOD, Department of CSE), [3]Associate Professor (Department of CSE)

Abstract - License plate recognition (LPR) technology is a mature yet imperfect technology used for automated toll collection and speed enforcement. The portion of license plates that can be correctly recognized and matched at two separate stations is typically in the range of 35% or less. Existing methods for improving the matching of plates recognized by LPR units rely on intensive manual data reduction, such that the misread plates are manually entered into the system. Recently, an advanced matching technique that combines Bayesian probability and Levenshtein text-mining techniques was proposed to increase the accuracy of automated license plate matching. The key component of this method is what we called the association matrix, which contains the conditional probabilities of observing one character at one station for a given observed character at another station. However, the estimation of the association matrix relies on the manually extracted ground truth of a large number of plates, which is a cumbersome and tedious process. To overcome this drawback, in this study, we propose an ingenious novel RLLPR-NED (Recursive Learning License Plate Recognition-Normalized Edit Distance) algorithm that eliminates the need for extracting ground truth manually. These automatically learned association matrices are found to perform well in the correctness in plate matching, in comparison with those generated from the painstaking manual method. Furthermore, these automatically learned association matrices outperform their manual counterparts in reducing false matching rates. The automatic RLLPR-NED method is also cheaper and easier to implement and continues to improve and correct itself over time.

Keywords - RLLPR-NED (Recursive Learning License Plate Recognition with Normalized Edit Distance), text mining, vehicle tracking.

1. INTRODUCTION

In April 2006, Knoxville, Tennessee joined an increasing number of cities in reducing speed limits for large trucks (with gross weights over 10,000 pounds) on the interstate highways in its metropolitan area. While reducing truck speed limits is a relatively simple act for metropolitan planning agencies, the enforcement side of it often meets with more challenges. This is the case for Tennessee Highway Patrol (THP), which has jurisdiction over Interstates 40 and 75 (I-40 and I-75), which both pass through the Knoxville metropolitan area. After the new speed law was enacted, THP found itself facing 12 million large trucks, most of which were going faster than the 88 km/hour (55 mph) speed limit, on this stretch of interstate; a complicating issue for the THP was that it was provided no annual budget or manpower increases for the purpose of enforcement. To this end, the University of Tennessee proposed the use of license-plate recognition (LPR) equipment to automatically track large trucks as they cross through the metropolitan area. This system would function in real time without the need for mailing out speeding tickets after the fact or pulling trucks over after dangerous high-speed pursuit, both of which alternatives are resource and labor-intensive. License-plate recognition technology was originally developed to read license-plate characters on moving vehicles. The process of capturing a plate image and recognizing the characters involves vehicle detection, image processing, and optical character recognition, which have all been documented in detail in past studies. The concept of an LPR-based speed enforcement system is alluring: with simple installation of LPR units, real-time truck monitoring seems easily attainable. The reality is not so simple, and, hence, the potentialities of LPR are not quickly realizable. Depending on the type of internal technology, the installation, the on-site calibration, the weather, the lighting, the plate configuration, and a host of other conditions, LPR rarely recognizes more than 80% of the plates and often does worse than 60%. Fortunately, all is not lost; even when LPR fails to read a plate, meaning that not every single character is recognized correctly, the system usually returns very valuable and mostly correct individual character information. By comparing the imperfectly read plate against another such plate, one may still be able to render reasonable judgment in terms of whether the two plates are actually a match. For instance, if two strings (sequence of characters) differ from each other by only one character, they may well have originated from the same plate. Therefore, a measure of similarity between two strings can be established to indicate the likelihood of a match.

2. SYSTEM MODEL

License plate recognition (LPR) method is a mature yet justification method used for automated toll data collection and speed enforcement. Because of LPR's limited accuracy, around or less than 60%, depending on the model, installation, variation of the license plates in the traffic stream, and other factors, the portion of license plates that was suitably recognized and matched at two distinguish nodes was typically in the range of 35% or less. This number further deteriorates if one tries to match the same plate through more than two sequential stations. Existing techniques for improving plate matching accuracy consist of intensive manual data reduction and posterior training [1], [2]. Through the study of the characteristics of the errors made by LPR hardware, we can find out that, while significant portion of plates were recognized incorrectly, many of these misread plates only had one, two, or three misread characters out of the entire string of six or so total characters, as shown in figure 1.

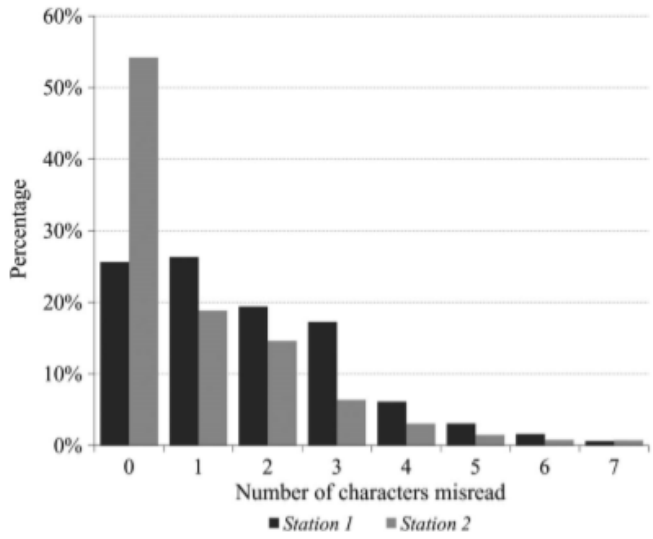

Figure 1: Sample LPR character-reading error rates per plate for two units installed at two different locations.

In other words, the accurate justification rate of individual words is much higher than the correct recognition rate of entire plates. This is a simple yet powerful fact unexplored by hardware manufacturers and researchers in the area of LPR technology and video image processing in general. Our idea takes advantage of this simple fact and explores the likelihood that two seemingly different license plate strings (sequence of alphanumeric characters) resultant from two LPR stations are actually a match. It should be pointed out that the license plate matching is far more challenging than a traditional template matching problem because license plate strings typically

(1) Do not have a readily available dictionary to compare to,
(2) Do not have a context to help "guess" the meaning of the plate, and
(3) Include both alphabetical and numerical characters having much possible syntax. In an automated license plate matching process, we do not know whether each plate string is recognized correctly at all. However, we still have to try to discern whether two strings, both could be incorrectly recognized, is a match.

3. PREVIOUS WORK

Measuring similarity of strings is a well-known problem in computer science which has applications in many fields such as computational biology, text processing, optical character recognition, image and signal processing, error correction, information retrieval, pattern recognition, and pattern matching in large databases.

License plate recognition (LPR) method is suppurate even fallible method used for machine-driven toll collection and speed enforcement [1]. The section of license plates that can be right recognized and mates at two distinguish nodes is typically in the valid range of 45% or less [1]. Dictionary lookup methods are popular in dealing with ambiguous letters which were not recognized by Optical Character Readers [2]. However, a robust dictionary lookup method can be complex as apriori probability calculation or a large dictionary size increases the overhead and the cost of searching [3]. Assume that two character array X and Y over a constant alphabet, the normalized edit distance between X and Y, d(X, Y) is explained as minimum as of W(P)/L(P), where P is an editing way between X and Y, W(P) is the addition of weights value of the traditional edit operations of P, and L(P) is the count of these operations [3]. Generally, d(X,Y) cannot be evaluated by initial obtaining the un-normalized edit distance between X and Y and then normalizing this cost by the length of respecting editing path [3]. Markov graph learning supports lower limits on the counts of samples accepted for any procedure to learn the Markov graph outline of a probability distribution, up to edit distance [4]. An extremely-used symbol of character array similarity is the edit distance: The limited count of insertions, deletions and substitutions essential to convert one character array into other [5].

4. PROPOSED METHODOLOGY

We propose a novel RLLPR-NED (Recursive Learning License Plate Recognition with Normalized Edit Distance) algorithm that can generate these important association matrices without the need of extracting initial truth manually. These self driven learned association matrices are searched to perform well in the decidedness in license plate matching, in comparison with those represented from the painstaking manual technique. Generally, these recursive learned association matrices represents their manual counterparts in decreasing non true matching rates. The self driven RLLPR-NED technique is suitable and easier learn to implement and continues to increase performance and accurate itself over time.

The proposed method RLLPR-NED is frequently used to intelligence transportation system. The process of matching two strings involves a sequence of comparisons of individual characters to determine the degree of similarity between the two. Consider, for example, a license plate with the string “4455HZ” which is read by two LPR machines at two different locations. Suppose that at the first location, the plate was read as “4455IIZ” and at the second, “4455HZ.” Note that neither LPR unit “knows” whether it has read the plate correctly. By looking at the two reports, one can either declare no match, or one may perhaps speculate a potential match since the two strings differ only by two pairs of characters: “I”-“H” and “I”-“” (where “” represents a null or empty character). If there were another plate that was read as “445OHZ” earlier at the first location, one may speculate that it is less likely that the “O”-“5” pair is a match. The task here is to “teach” the computer to make such speculations. Techniques for measuring the similarity or dissimilarity between two strings have been developed in the past and have found application in areas such as handwritten character recognition and computation biology. The pioneer in this field is Vladimir Iosifovich Levenshtein, who developed Edit Distance (ED) (also known as Levenshtein distance), which is a metric that computes the distance between two strings as measured by the minimum-cost sequence of edit operations. Given two strings x and y, their Edit Distance describes how many fundamental operations are required to transform x into y. These fundamental operations are termed as follows:

- Substitutions: A character in x is replaced by the corresponding character in y.
- Insertions: A character in y is inserted into x, thereby increasing the length of x by one character.
- Deletions: A character in x is deleted, thereby decreasing the length of x by one character.

To relate the definition of Edit Distance to the problem presented in this dissertation work, we will return to the example of the plate "4455HZ" being captured by two LPR stations. Let x = "4455IIZ" and y = "4455HZ"; the task is to compute the number of fundamental operations to transform x into y. (Note that x and y could have been assigned in reverse order since the “true” plate number is not known.) In this case, it can be established that the minimum number of operations is 2, which corresponds to the substitution of the first “I” in x by “H” and the deletion of the second “I” in x. Therefore, the Edit Distance d(x,y) between x and y is 2. To determine the minimum d(x,y) for any pair of strings x and y there efficient computational procedure called dynamic programming. In many applications, the string y is provided by a list of words that has the maximum likelihood of containing the “true” value of the given string, x. This pre-specified list of words is called a lexicon or reference for matching. Using this list of words, it is possible to detect errors, generate candidate corrections, and rank these candidates. However, in the problem presented in this paper, x and y represent strings read from license plates and either one can be used as a reference for matching since there is no knowledge about the true string.

The normalized edit distance has been explained here implemental in terms of paths rather than edit transformations. Suitably, unless certain non-trivial situations are imposed on the traditional edit weight function y and on the concept of edit sequences, no meaningful explanation of normalized edit distance seems suitable in terms of edit transformations. For example, if y is zero for desired pairs of elements, then for any two character array X, Y, there could be undetermined long sequences of traditional edit operations with normalized weight is equal zero. On the other way, it should be noticed that the minimization can by no means be accepted out by first minimizing W(P) through and then normalizing it by the length value of the obtained path.

We devise a recursive-learning algorithm to eliminate the need for painstakingly deriving truth matrices, which relies on human verification of plate strings. The association matrices are automatically learned through a self-learning algorithm. The automated learning algorithm takes into the consideration that, while the suitable justification rate of a plate string is relatively low, usually 30%–50% without calibration, the accurate justification rate of individual characters is much higher level. That is, when a license plate is not to be accurately read, there is remains quite a lot of desirable and suitable information embedded in the inaccurate character array. With this as the departure point, the algorithm starts with a clean association matrix and

begins a self-learning process with every new plate string reported from an LPR unit and continues to learn over time.

5. IMPLEMENTATION TOOLS

Two survey periods over the LPR system operation were selected to assess the performance of our proposed self-learning algorithm. At the first survey period (five complete days of operation in 2010, i.e., April 6th and 7th and May 25th, 26th, and 27th), both LPR readings and their ground truths were obtained. As for the second survey period (41 days of operation in April and May of 2010), only the plate readings were obtained. The first data sample, containing ground truth sets, is the reference sample for validation of the self-learning algorithm. The second sample is used to estimate a sequence of association matrices using the recursive-learning procedure. This text mining approach is implemented through MATLAB (MATrix LABoratory) software.

MATLAB is a large-performance, effective and interactive language for essential computing approach. It combines computation, visualization, graphical and programming in an easy environment where questions and solutions are solved in familiar mathematical notation and graphical way. It include mathematical matrix form of image and other computational procedure development data acquisition modeling, picture processing, information processing, simulation and prototyping data processing, visual scientific, engineering drawing and graphics application development consisting graphical programmer interface area building MATLAB (Matrix Laboratory) is an effective programming way whose basic information node is an array in multiple dimensional plan, which does not consider to mention dimensioning. This permits you to prove various technical problems in various formats, especially those with matrix form and vector calculations. In a few durations it would assume to write a code in a desired scalar language such as C or FORTRAN. MATLAB is stands for matrix laboratory. MATLAB was generally written to perform easy access to matrix software implemented by the LINPACK, EISPACK and various technical projects. Today, MATLAB compiler supports to incorporate the LAPACK packages, embedding the suitable in software for matrix evaluation and programming issues**.** MATLAB is used in every level of computational mathematics. Following are some desired used mathematical implementations where it is used most commonly:

- Dealing with Matrices and Arrays
- 2-D and 3-D Plotting and graphics
- Linear Algebra
- Algebraic Equations
- Non-linear Functions
- Statistics
- Data Analysis
- Calculus and Differential Equations
- Numerical Calculations
- Integration
- Transforms
- Curve Fitting
- Various other special functions

MATLAB has performed over duration of years with multiple inputs from various programmers. In university research area, it is the benchmark and effective instructional method for introductory and new generation courses in engineering and medical science field. In engineering field, MATLAB is the tool of select for best large-productivity research concern, development and analysis. MATLAB support basic key points a family of application depend solutions called toolboxes. Following are the basic features of MATLAB:

- It is a high-level language for mathematical computation, computer visualization and application development.
- It support a suitable environment for repetitive exploration, outline and problem objective solving.
- It provides library of numerical functions for algebra, statistics, Fourier analysis, filtering, optimization, integration and solving ordinary differential equations.
- It provides built-in graphics function for visualizing information and tools for creating custom plots.
- MATLAB interface provide development tools, GUI used for enhancing program quality and maintainability and modify maximum performance.
- It support techniques for making applications with graphical interfaces.
- It support mapping for MATLAB algorithms with external applications.

REFERENCES

[1] S. M. Turner, “Advanced techniques for travel time data collection”, Transp. Res. Rec., vol. 1551, pp. 51–58, 1996.

[2] “Travel Time Data Collection Handbook: Office of Highway Information Management, Federal Highway Admin., Washington, DC, USA, 1998. [Online]. Available: http://www.fhwa.dot.gov/ohim/start.pdf

[3] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals process", Physics Doklady, vol. 10, no. 8, pp. 707–710, 1966.
[4] F. M. Oliveira Neto, L. D. Han, and M. K. Jeong, "Tracking large trucks in real-time with license plate recognition and text-mining techniques", Transp. Res. Rec., vol. 2121, pp. 121–127, 2009.
[5] F. M. Oliveira Neto, L. D. Han, and M. K. Jeong, "Online license plate matching algorithm using license-plate recognition machines and new weighted edit distance", Transp. Res. Part C, Emerg. Technol., vol. 21, no. 1, Apr. 2012.
[6] A. Marzal and E. Vidal, "Evalutation of normalized edit distance and applications", IEEE Trans. Pattern Anal. Mach. Intell., vol. 15, no. 9, Sep. 1993.
[7] R. A. Wagner and M. J. Fischer, "String-to-string correction problem", J. Assoc. Comput., vol. 21, no. 1, pp. 168–173, Jan. 1974.
[8] R. O. Duda, P. E. Hart, and D. G. Stork, "Recognition with strings", in Pattern Classification, 2nd ed. Hoboken, NJ, USA: Wiley-InterScience, 2000, ch. 8, pp. 413–420.
[9] B. J. Oommen, "Constrained string editing", Inf. Sci., vol. 40, no. 3, pp. 267–284, Dec. 1986.
[10] T. Okuda, E. Tanaka, and T. Kasai, "A method for correction of garbled words based on the Levenstein metric", IEEE Trans. Comput., vol. C-25, 1976.
[11] G. Seni, V. Kripasundar, and R. Srihari, "Generalizing edit distance to incorporate domain information: Handwritten text recognition as a case study", Pattern Recognit., vol. 29, no. 3, 1996.
[12] J. Wei, "Markov edit distance", IEEE Trans. Pattern Analysis Mach. Intell., vol. 26, no. 3, 2004.
[13] Ristad and Yianilos, "Learning string-edit distance metric", IEEE Trans. Pattern Anal. Mach. Intell., vol. 20, no. 5, pp. 522–532, 1998.
[14] M. Bilenko and R. J. Mooney, "Adaptive duplicate detection using learnable string similarity measures", in Proc. 9th ACM SIGKDD Int. Conf. KDD, 2003.

AUTHOR'S PROFILE

Anand Chourey has received his Bachelor of Engineering degree in Computer Science & Engineering from Indira Gandhi Engineering College, Sagar in the year 2012. At present he is pursuing M.Tech with the specialization of Computer Science and Science in Millennium Institute of Technology and Science, Bhopal. His area of interest Data Mining, Image Processing etc.

Avinash Sharma has received his M.Tech from BUIT, Bhopal in the year 2010. At present he is working as an Associate Professor and also Head of CSE Department at Millennium Institute of Technology and Sciences, Bhopal. His areas of interests are Data Mining, Image Processing, Cloud Computing etc.

Madhuvan Dixit has received his M.Tech from RGPV, Bhopal in the year 2014. At present he is working as an Assistant Professor in Millennium Institute of Technology and Sciences, Bhopal. His areas of interests are Data Mining, Image Processing, Cloud Computing, Grid Computing, Video Processing etc..