Error Classification in OCR Historic Newspaper Archive using multi-class Support Vector Machine

Megha Gupta and Haimonti Dutta*
Department of Computer Science, IIIT-Delhi
**Affiliated to The Center for Computational Learning Systems, Columbia University, New York
(meghag, haimonti)@iiitd.ac.in

Abstract

Optical Character Recognition (OCR) is a commonly used method of digitizing printed texts so that they can be searched and displayed online, stored compactly and used in text mining applications.
The text generated from OCR devices, however, is often garbled due to variations in quality of the input paper, size and style of the font and column layout. This adversely affects retrieval effectiveness; hence the techniques for cleaning the OCR need to be improvised. Often such techniques involve laborious and time consuming manual processing of data.
This paper shows the need to devise an algorithm that is scalable for a large dataset. The current state of the art algorithm used for performing multi class classification is not yet scalable. The current algorithm takes a long time to converge in a particular parameter setting.

1 Introduction

The *California Digital Newspaper Collection*[1] is an initiative of the Center for Bibliographical Studies and Research (CBSR) which is supported in part by the U.S. Institute of Museum and Library Services and National Endowment for the Humanities (NEH) to digitise California newspapers for the National Digital Newspaper Program (NDNP). It contains over 400,000 pages of significant historical California newspapers published from 1846-1922.
OCR devices are widely used in electronic conversion of scanned images which are handwritten or printed text into a machine encoded text. The scanning generates It finds most successful applications in the field of machine Learning, Artificial Intelligence and Pattern recognition. It deals with the problem of recognising optically generated characters be it offline or online. The performance directly depends on the quality of input document. The more constrained the input is the better will be the performance of the system. But when it comes to unconstrained handwriting, the performance is far from satisfactory. The main application areas for (Eikvil 1993) like Automatic number plate readers, form readers, signature verification.

[1]http://cdnc.ucr.edu/cgi-bin/cdnc

This project deals with printed text in the form of Historical Newspaper Articles in the holdings of (Collection 2009). One such newspaper, The Amador Ledger published in the early 1900s by the Amador Publishing Company appealed to the community's interests by covering issues unique to gold mining. Patrons of the (Collection 2009) continue to be interested in studying about the status of the local mining industry and consequently read the Amador Ledger on a regular basis even to this day and correct (Eikvil 1993) errors as they come across them.
In this paper, we perform error classification using Joachims multi-class Support Vector Machine algorithm called $SVM^{multiclass}$[2]. We chose this algorithm as its the state of the art algorithm till now. But the experiments give altogether a different view on this algorithm. This algorithm do not converge quickly on certain parameters which are shown in table 4

2 Related Work

OCR correction

The OCR enables searching of full text data but it is not 100% accurate. Its accuracy largely depends on paper quality of the original issue, font style, column layouts, its condition at the time of microfilming, choice of scanner (Jeffrey Esakov and Zhou), and the quality of the OCR software. The OCR errors are widely classified as non-word errors and real-word errors. A non-word error takes place when the erroneous word is not a dictionary word whereas in real-word error the erroneous word is correctly spelt and therefore can be found in the dictionary but incorrectly used in context. A related work on the OCR error classification is done by (Da?ason 2012). It gives a more exhaustive classification with respect to OCR errors that helps in classifying real words not contained in any of the dictionary, e.g. names, out-dated or historic spelling variations. Segmentation errors, Hyphenation error, Misrecognition of characters, Punctuation errors, Case sensitivity, Changed word meaning are the classes required for better classification of OCR errors.

[2]http://www.cs.cornell.edu/people/tj/svm_light/svm_multiclass.html

Multi-class SVM

SVMs are inherently binary classifiers. To solve the multi class problem, construction of several binary classifiers is required and the result from each classifier is combed. The traditional approaches to do multi class classification are: a) *one-versus-all (OVA classification)* classifies the new instances is done by winner-takes-all strategy where the classifier with highest output function assigns the label. It generates k classifiers for a k class problem. b)*one-versus-one* uses the majority voting strategy where each classifier assigns the new instance one of the two class, thereby increasing the vote of the assigned class. Finally the class with the majority votes determines the instance classification. It generates k(k-2)/2 models for a k class problem. This approach is not practical for large-scale classification because of the memory required for storing k(k-1)/2 models. c)*Directed Acyclic Graph SVM* d)*error correcting output codes*
The other alternative is by using *pairwise classification* (Brunner et al. 2012), where a two class classifier is build on the two input example. The class of training example might be unknown but the mandatory condition is to know whether the examples belong to the same class or not. The input pair is positive pair when the both belong to the same class, otherwise its called as negative pair.
The (Crammer and Singer 2002) approach was to pose the multi class classification problem into a single optimization problem, rather than decomposing it into multiple binary classification problems.

3 The Data

Newspaper

Figure 1 shows a scanned newspaper (*The Amador Ledger*, January 26,1900) from the (Collection 2009) archive highlighting the article to be corrected by the user. The raw OCR text of the highlighted article is displayed by clicking on the specific article headline from the list of headlines seen on the left side of the issue in figure 1.

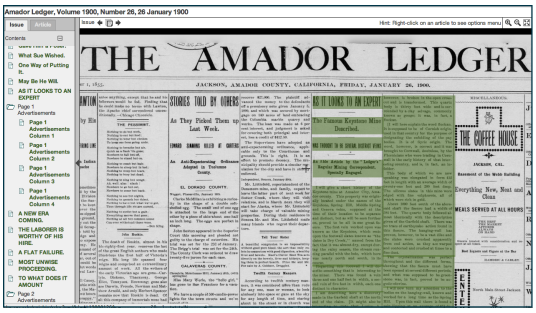

Figure 1: Amador Ledger

The tool used by patrons to correct OCR text is shown in figure 2. All the corrections performed by the annotators were recorded in the log files.

The text correction statistics consisting of total number of annotators involved in the the data enrichment process are shown in figure 3. The total number of lines corrected by 848 users are 1,705,149.

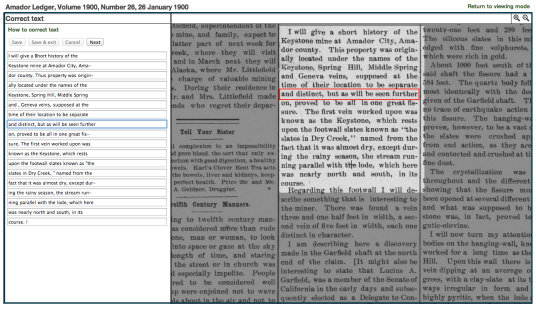

Figure 2: Text correction tool

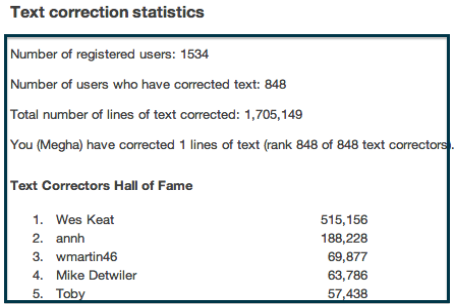

Figure 3: Text correction statistics

Log-files

The log files are the files that maintain the history of raw OCR and its corresponding corrected OCR text. These files were generated using a third party software issued by *Veridian*[3], a digital library software. Each of the log file was generated at an issue-level, containing xml data about the multiple pages in the issue. The table 1 describes the structure and the format of the log file. The following information is provided about the corrections made by the patrons:
a) *Page Id* representing the id of the page in which editing was done. b) *Block Id* representing the id of the paragraph containing the line corrected by the user. c) *Line Id* is the id of the particular line edited by the user. d) *Old text* is the garbled text generated by the OCR device and replaced by the user. e) *New text* is the corrected text with which the old text was replaced.

<TextCorrectedLine lineID="P2_TL00800">
<OldTextValue>**Spll, Stales**</OldTextValue>
<NewTextValue>**Union Stables**</NewTextValue>
</TextCorrectedLine>
<TextCorrectedLine lineID="P2_TL00801">
<OldTextValue>**?** Under Webb Hall *</OldTextValue>
<NewTextValue>**Under Webb Hall** </NewTextValue>
</TextCorrectedLine>

Table 1: A segment of the log file

There were in total 235 log files of which we worked on

[3] http://www.dlconsulting.com/

191.

Preprocessing

In order to make use of the full text data, we did some preprocessing on the data stored in the log files. We extracted the old text and the corresponding new text from the log file and tokenized them into words. These tokens were separated by whitespaces. There were in total 44,023 tokens of which 21,113 were corrected by the annotators. The errors rectified by the users were categorized as spellcheck error , addition of a new word, capitalization error, typo, punctuation error. The distribution of the these classes in the dataset is shown in figure 4

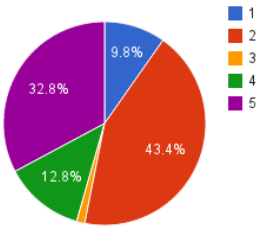

Figure 4: Error Classification

4 Methodology

We are performing automatic text correction by modelling it through a machine learning algorithm, called multi class Support Vector Machines (Joachims 2008). In this formulation, multi class problem is posed as a constrained optimization problem with a quadratic objective function. The multi class formulation is stated as

$$min1/2\sum_{i=1}^{k} w_i * w_i + C/n\sum_{i=1}^{n}\xi_i \quad (1)$$

$$s.t.\forall y \leq k : [x_1 \cdot w_{yi}] \geq [x_1 \cdot w_y] + 100 * \Delta(y_1, y) - \xi_1$$

. . . . . .

$$s.t.\forall y \leq k : [x_n \cdot w_{yn}] \geq [x_n \cdot w_y] + 100 * \Delta(y_n, y) - \xi_n$$

Here C is the regularization parameter that trades off margin error and training error. $\Delta(y_1, y)$ is the loss function that returns 0 if y_n equals y, and 1 otherwise. ξ_i are the non negative slack variables which measure the degree of misclassification of the data x_i.
(Joachims 2008) has to two modules, *svm_multiclass_learn* and *svm_multiclass_classify*. The learning module learns the model given the parameters and the training data whereas the classification module applies the learned model to the test data to find the error. When the data is not linearly separable, the original finite dimensional space is mapped to a much higher dimensional space in order to make the separation easier in that space. The mappings used by SVM schemes are designed to ensure that the dot products may be computed easily in terms of the variables in the original space, by defining them in terms of a Kernel function selected to suit the problem. The types of kernel function used can be described as follows:
a) Linear Kernel (default) : It is the basic kernel function given by the inner product <x,y>plus an optional constant c. K(x,y) = x^Ty + c
b) Polynomial Kernel : It is a non-stationary kernel which is well suited for problems where the training data is normalized.
K(x,y) = $(\alpha x^T y + c)^d$
c) Radial Basis Kernel : The (Gaussian) radial basis function kernel on two samples x and y represented as feature vectors in some input space is defined as
K(x,y) = exp($-||x - y||^2/2\sigma^2$)

5 Empirical Evaluation

Preprocessing & Data Generation

Feature Construction Originally, there were two features in the dataset, that is old text and new text. Further features were manually crafted looking at the types of errors. In our dataset, we have six binary features consisting of SameLength, EditDist_0, EditDistAbove1, EditDistBelow3, EditDist_1andCaseChange, PunctuationDiff.

1. "SameLength" is 1 if both the old word and new word have same length
2. "EditDist_0" is 1 if both the words are exactly same
3. "EditDistAbove1" is 1 if more than one edit operation is required to convert old word to new word
4. "EditDistBelow3" is 1 if less than three edit operations are required to convert old word to new word
5. "EditDist_1andCaseChange" is 1 if the two words have edit distance is exactly 1 and the first character of one string change from upper case to lower case or vice versa.
6. "PunctuationDiff" is 1 if both the old text and new text differ in any of the following punctuation marks !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~

Label Construction The error classes were restricted to 6 classes including Spellcheck Error, Addition of a new word, Capitalization Error, Typo, Punctuation Error and No correction These labels were assigned according to the flow graph as shown in figure

1. Spellcheck error : When the edit distance is between 1 and 3. For example, mounten and mountain.
2. Addition of a new word : When the edit distance is more than 3. For example, at and attend.
3. Capitalization error : When the edit distance of two strings is exactly 1 and first letter of both the strings changes from upper to lower case or vice versa. For example, largest and Largest.
4. Typo : When the edit distance is exactly one and case change is 0. For example, teh and the

5. Punctuation error : When the two strings differ by special characters contained in the set (!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~). For example, "residents" and residents
6. No correction : When the old and new text are same. For example, plant and plant

The dataset was parsed to a format used by the Joachim's multi class SVM algorithm which is represented as
<target> <feature>:<value>.....<feature>:<value>
The number of rows in the dataset is 44,022. The class distribution in the dataset is shown in table 2

$Class$	$no.of instances$
1	1970
2	8732
3	261
4	2572
5	6602
6	23885
Total	44022

Table 2: Class Distribution

Experiment Setup

Experiments were performed by randomly partitioning the data into 70% (30,815 instances) and 30% (13,207 instances) of training and testing data respectively. For each experiment, the regularization parameter (C) and the type of kernel (t) was varied to (0.001,0.01,0.1,1,10,100) and (Linear, Polynomial and Radial basis kernel) respectively. The experiments were performed on two machines, one of which was a linux server and the other was a dual core Mac machine with Intel Core i7 processor, 8GB RAM, 2.9 GHz of processor speed. Each experiment was ran for 5 iterations. The average loss on the test set and CPU runtime were noted to analyse the experiments. Table 3 describes how the average loss on test set varies for different values of 'C' and 't'. Here, AE_L, AE_P, AE_R describes the average loss error of linear, polynomial and rbf kernels respectively. Table 4 shows the average runtime (cpu sec) required to learn a model for various values of C and for different kernels. Here, AT_L, AT_P, AT_R refers to the average run time of linear, polynomial and rbf kernels respectively.

Results

C	AE_L	AE_P	AE_R
.001	32.06 $\pm$0.149	32.06$\pm$0.149	11.96$\pm$0.101
.01	32.06$\pm$0.149	32.06$\pm$0.149	12.04$\pm$0.199
.1	32.06$\pm$0.149	29.10$\pm$0.133	12.04$\pm$0.199
1	29.10$\pm$0.133	28.99$\pm$0.264	6.36$\pm$0.142
10	29.20$\pm$0.133	8.338$\pm$0.107	6.36$\pm$0.142
100	29.20$\pm$0.133	0.58$\pm$.034	6.35$\pm$0.16

Table 3: Experiment Results : Average loss error (test set)

C	AT_L	AT_P	AT_R
.001	0.218	5192	4596
.01	0.202	5166	4460
.1	0.208	72225	4377
1	0.678	81251	16468
10	0.672	260316	55308
100	0.998	106753	48734

Table 4: Experiment Results : Training time (cpu sec)

Discussion

As can be seen in table 3 , the error is minimum when the C = 100 and when the kernel type is polynomial. But the time taken as shown in table 4 to train the model for these parameters, when C is 100 is 106753 cpu sec.

Known Defects

The (Joachims 2008) algorithm converges quickly for linear type kernel but the its performance on test set is poor. For non linear kernels, this algorithm does not scale well for large scale datasets but gives better performance than linear kernels.

6 Conclusion & Future Work

The California Digital Newspaper Collection has an archive of 400,000 pages of historical California newspapers published between 1846 to 1922. This archive which has been subjected to OCR and is currently stored in an online database making them accessible to patrons. The OCR scanning process generates lot of garbled text which needs to be corrected to make the online newspaper repository more accessible to general public. An automatic OCR error correction technique is implemented using machine learning technique, multi class Support Vector Machine used to build a model for predicting errors in the dataset generated. State-of-the-art algorithm (Joachims 2008) was used to for our experiments to predict the errors. The labels generated using the approach mentioned in section 5 served as the ground-truth against which labels generated by the algorithm were compared. Our results indicate that even the state-of-the-art algorithm is not scalable for large scale learning as it takes 106753 cpu sec to learn a model on data with 30,185 instances. Our Future work involves careful analysis of the current algorithm to scale the non-linear kernels for large scale learning.

Acknowledgment

This work is supported by funding from the National Endowment for Humanities. I would also like to express my gratitude to my Advisor, Dr. Haimonti Dutta for her support, patience and encouragement throughout the research.

References

Anand Rajaraman, Jure Leskovec, J. D. U. 2013. *Mining Of Massive Datasets*. Cambridge University Press.

Brunner, C.; Fischer, A.; Luig, K.; and Thies, T. 2012. Pairwise support vector machines and their application to large scale problems. *J. Mach. Learn. Res.* 13(1):2279–2292.

Carlson, A. J.; Rosen, J.; and Roth, D. 2001. Scaling up context-sensitive text correction. *Proceedings of the Thirteenth Conference on Innovative Applications of Artificial Intelligence Conference* 45–50.

Collection, C. D. N. 2009. http://cdnc.ucr.edu/cdnc.

Crammer, K., and Singer, Y. 2002. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res.* 2:265–292.

Da?ason, J. F. 2012. *Post-Correction of Icelandic OCR Text*. Faculty of Industrial Engineering, Mechanical Engineering and Computer Science, University of Iceland.

Eikvil, L. 1993. Optical character recognition.

Gao, J.; Li, X.; Micol, D.; Quirk, C.; and Sun, X. 2010. A large scale ranker-based system for search query spelling correction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, 358–366. Stroudsburg, PA, USA: Association for Computational Linguistics.

Hoi, S. C.; Jin, R.; and Lyu, M. R. 2006. Large-scale text categorization by batch mode active learning. In *Proceedings of the 15th international conference on World Wide Web*, 633–642. ACM.

Huang, G. B.; Kae, A.; Doersch, C.; and Learned-Miller, E. 2012. Bounding the probability of error for high precision optical character recognition. *J. Mach. Learn. Res.* 13(1):363–387.

Jeffrey Esakov, Daniel P. Lopresti, J. S. S., and Zhou, J. Issues in automatic ocr error classification.

Joachims, T. 1998. Text categorization with support vector machines: Learning with many relevant features. In Ndellec, C., and Rouveirol, C., eds., *Machine Learning: ECML-98*, volume 1398 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 137–142.

Joachims, T. 1999. Advances in kernel methods. Cambridge, MA, USA: MIT Press. chapter Making large-scale support vector machine learning practical, 169–184.

Joachims, T. 2008. Multi-class support vector machine.

Kukich, K. 1992. Techniques for automatically correcting words in text. *ACM Comput. Surv.* 24(4):377–439.

Menon, A. K. 2009. Large-scale support vector machines: algorithms and theory. *Research Exam, University of California, San Diego*.

Niklas, K. Unsupervised post-correction of ocr errors.

Niklas, K. 2010. Unsupervised post-correction of OCR errors. Master's thesis, Leibniz-Universität Hannover, Hannover, Germany.

Reynaert, M. 2008. Non-interactive OCR post-correction for giga-scale digitization projects. In Gelbukh, A., ed., *Computational Linguistics and Intelligent Text Processing*, volume 4919 of *Lecture Notes in Computer Science*. Berlin/Heidelberg: Springer. chapter 53, 617–630.

Tong, X., and Evans, D. A. 1996. A statistical approach to automatic ocr error correction in context. In *Proceedings of the Fourth Workshop on Very Large Corpora (WVLC-4*, 88–100.

Yamangil, E., and Nelken, R. 2008. Scalable lexical correction from wikipedia edits using perceptron reranking. *unpublished*.