Classification of Crowdsourced Text Correction

Haimonti Dutta
Center for Computational Learning Systems, Columbia University, New York, NY 10115
haimonti@ccls.columbia.edu

Megha Gupta
Dept. of Computer Science IIIT-Delhi, India.
meghag@iiitd.ac.in

Brian Geiger
Center for Bibliographical Studies and Research
University of California, Riverside
brian.geiger@ucr.edu

***Abstract*—**

Optical Character Recognition (OCR) is a commonly used technique for digitizing printed material enabling them to be displayed online, searched and used in text mining applications. The text generated from OCR devices 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 and hence techniques for cleaning the garbled text need to be improvised. Often such techniques involve laborious and time consuming manual processing of data. This paper presents a prototype system for Classification of Crowdsourced Text Correction (CCTC) which takes as input log files containing garbled and manually corrected OCR text, parses and tokenizes them and builds models for categorizing the corrections using state-of-the-art machine learning algorithms. Retrieval effectiveness on the California Digital Newspaper Collection is measured using Spearman's rank correlation metric. This prototype system is expected to be deployed on historical newspaper archives that make extensive use of user text corrections.

***Keywords*-component; formatting; style; styling;**

I. Introduction

Crowdsourcing is used extensively in cultural heritage and digital history related projects in recent years to digitize, create and process content and provide editorial or processing interventions. For example, the Australian Newspapers Digitization Program [1] allows communities to explore their rich newspaper heritage by enabling free online public access to over 830,000 newspaper pages containing 8.4 million articles. The public enhanced the data by correcting over 7 million lines of text and adding 200,000 tags and 4600 comments [2]. FamilySearch [3] made available handwritten digital images of births, deaths and marriage records for transcription by the public. The New York Public Library has 1,277,616 dishes transcribed to date from 17,079 menus.

In all of the above crowdsourcing projects, large volumes of data are generated by users. These include tags, folksonomies, flagged content, information on history, relationship and preference data, structured labels describing objects and creative responses [4]. However, little statistical analysis is done of the user generated content in most cases. Assessment of data quality obtained by leveraging the "wisdom of the crowd" remains an open problem.

In this paper, we focus on understanding the nature of text corrections done by users of an old historic newspaper archive. The newspapers are made available for searching on the Internet after the following processes take place: (1) the microfilm copy or paper original is scanned; (2) metadata is assigned for each page to improve the search capability of the newspaper; (3) OCR software is run over high resolution images to create searchable full text. The OCR scanning process is far from perfect and the documents generated from it contains a large amount of garbled text. A user is presented with a high resolution image of a newspaper page along with erroneous or distorted text from the OCR and is expected to rectify the garbled words as appropriate. A prototype for a system that can be used for **C**lassification of **C**rowdsourced **T**ext **C**orrection (CCTC) is presented which can answer simple questions such as "What are the different kinds corrections proposed by users?" and provide statistics generated from the correction process. The output from the system can be used to enhance search and retrieval.

The study used log files generated from text correction software in use at the California Digital Newspaper Collection (CDNC)[1], which contains over 400,000 pages of newspapers published in California between 1846-1922. To the best of our knowledge, this is the first attempt to statistically analyze and model OCR error corrections provided by the crowd. We posit that such a classification system will be beneficial when attempting to compensate the annotators; it can also be used for task allocation if some users are more comfortable with certain type of corrections than others.

Organization: Section II describes the architecture of the proposed system; Section III presents empirical and scalability results on real-world data collected at CDNC; Section IV presents information retrieval techniques; Section V discusses related work. Finally, Section VI concludes the paper.

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

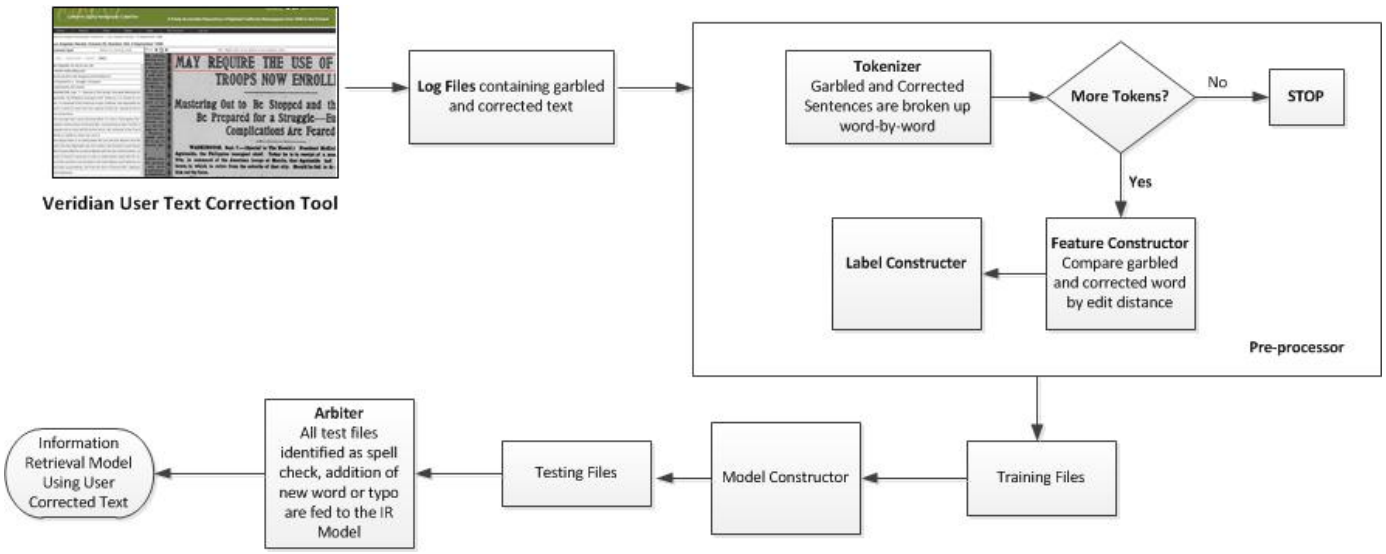


Figure 1. The Architecture of the Proposed System

II. Architecture of the Proposed System

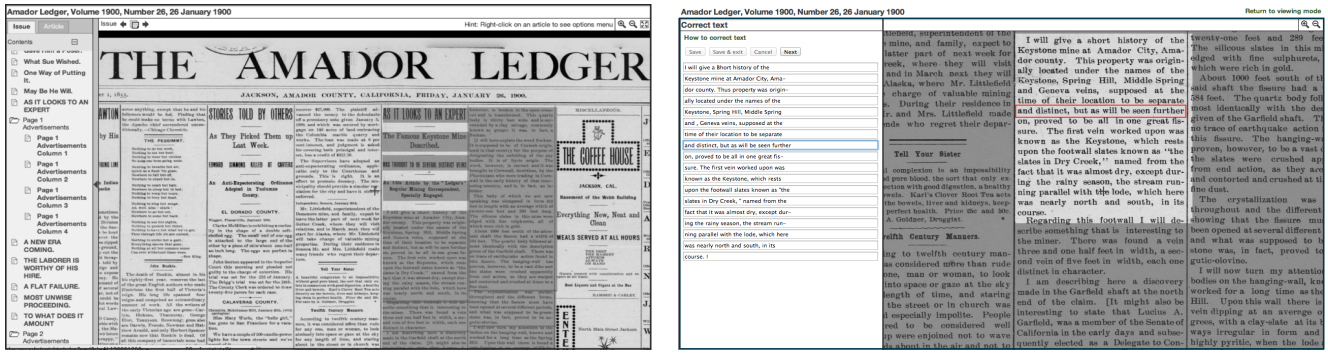
The **C**lassification of **C**rowdsourced **T**ext **C**orrection (CCTC) system has the following components:

1) The **Veridian User Text Correction**[2] tool which takes as input a scanned page of the newspaper and enables users to correct OCR errors as they come across them. Figure 2(a) shows an example of a scanned page from "The Amador Ledger" published on January 26, 1900. The article to be corrected by a user is highlighted. The raw OCR text from the article and the tool used by patrons to correct text is shown in figure 2(b).
2) **Log Files:** All corrections performed by the annotators are recorded in log files. To date approximately 1,705,149 lines have been edited by 848 annotators which resulted in 235 log files. A sample of 191 files has been used for this work. Each log file is generated at the issue-level and contains XML data about the pages in the issue. Table I describes the structure of the log file. The following information is provided about the corrections made by the patrons: (a) *Page Id:* The id of the page in which editing was done. (b) *Block Id:* The id of the paragraph containing the line corrected by the user. (c) *Line Id:* The id of the line edited by the user. (d) *Old Text Value* is the garbled text generated by the OCR device and replaced by the user. (e) *New Text Value* is the corrected text with which the old text was replaced.
3) **Preprocessor:** The preprocessor has three main components:
   - **Tokenizer:** The old text and the corresponding new text from the log file is tokenized by white-spaces. There are 44,022 tokens of which 21,108 are corrected by the annotators.
   - **Feature Constructor:** Features are crafted by computing the Levenshtein edit distance between the old word and its correction. The Levenshtein edit distance [5] is defined as the minimum number of single edit operations (insertions, deletions and substitutions) required to convert one string into another. Six binary features are generated as follows: (a) **Difference Length Zero** : 1, if both the old word and new word have same length and 0 otherwise. (b) **Difference Length Above One** : 1, if the length of new word exceeds the length of old word and 0 otherwise. (c) **Edit Distance One**: 1, if single edit operation is required to convert old word to new word and 0 otherwise. For example, the feature is 1 for tokens "Under" and "under" or "the" and "them". (d) **Edit Distance Above One**: 1, if more than one edit operation is required to convert old word to new word and 0 otherwise. For "Spll," and "Union", value is 1 as more than one edit operation is required to convert from old to new token. (e) **Edit Distance is 1 and Case**

[2]http://veridiansoftware.com/crowdsourcing/

<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 I
A segment of the log file



(a) Scanned newspaper highlighting an article to be corrected by a user.

(b) The tool used by patrons to annotate articles.

Figure 2. The Amador Ledger, Jan. 26, 1900.

Change: 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. For example, for "Stales" and "Stables", the value is 0 as there is no case change. (f) **Punctuation Difference**: 1, if both the old text and new text differ in any of the following punctuation marks (!"#$%&'*+,-./:;<=>?@[\]^_'{|}~).

- **Label Constructor:** The errors rectified by the users are categorized as addition, deletion, punctuation error, capitalization error, and spellcheck error. Specifically, (a) **Addition:** When the length of new string exceeds the length of old given the difference is made by alphanumeric characters and the edit distance is above one. For example, "6RAVR" and "GRAVEL". (b) **Deletion:** When the length of old string exceeds the length of new given the difference is made by alphanumeric characters and the edit distance is above one. For example, "VVe" and "We". (c) **Punctuation:** When the difference in the length of strings is non-zero and they differ by special characters contained in the set (!"#$%&'()*+,-./:;<=>?@[\]^_'{|}~). For example, "Ladies!" and "Ladies". (d) **Capitalization:** When both the strings have equal length, edit distance is exactly 1 and first letter of both the strings change from upper to lower case or vice versa. For example, "largest" and "Largest". (e) **Spellcheck:** When the difference between string is above zero and the edit distance contributed by alphanumeric character is exactly one or when the strings have same length and the edit distance is one or above one irrespective of the involvement of special characters. For example, "the" and "them" or "hanger" and "banger".

The distribution of the these classes in the dataset is shown in Table II. It must be noted that by design tokens are always assigned to one class, although in principle it may be possible to assign them to multiple classes[3].

Class	No. of Instances
Addition	4575
Deletion	5024
Punctuation	6401
Capitalization	299
Spellcheck	4809
Total	21108

Table II
CLASS DISTRIBUTION

4) **Baseline Construction** In order to test the performance of the manually crafted features of the dataset, we generated automatic rules using Association Rule Mining (ARM). The tokenization of the old and new text in the log files was modified to incorporate the addition and deletion of text using a character, that is if the token was deleted by the user then the character "∼" was assigned to the new token whereas if the user added a new token as a part of correction then the character "+" was assigned to the old token. All the corrected tokens were further split into character level data with the tuple <old char, new char>replacing the previous token level notion of <old token,new token>giving us $58,963$ rows of data. Of $58,963$ rows, we encountered that $55,612$ rows were redundant. Hence, the number of unique instances was found to be 3351. Some of the major character to character corrections are o→e, b→h, u→n. Figure 3(a), 3(b) shows histograms where some of the character level corrections have taken place. The maximum number of automatic rules generated by ARM algorithm were 6631. The parameters of the algorithm were set to minimum confidence: 1.0E-4, delta: 1.0E-4, upper

[3]For e.g. a correction of "t$e" to "the" could be either a Spellcheck or a Punctuation Error correction but we assign it to Spellcheck

bound minimum support: 1.0, lower bound minimum support: 1.0E-4. The error labels are the same as that for token level data. (a) **Addition:** + → T (b) **Deletion:** A → ∼ (c) **Capitalization error:** l → L (d) **Punctuation error:** ! → t (e) **Spellcheck error:** o → e

5) **Model Construction:** The model for classifying crowdsourced text correction is built using a Multiclass Support Vector Machine algorithm [6]. Each training point belongs to one of k different classes. The goal is to construct a function, which given a new data point, will correctly predict the class to which it belongs. Different methods have been proposed in literature for solving the SVM multi-class classification problem. Some popular techniques include: a) *One-Versus-All (OVA) classification* Build k different binary classifiers; for the i^{th} classifier, let the positive examples be all points in class i and negative examples not in class i. Let $f(x) = \arg max_i f_i(x)$. b) *one-versus-one* uses the majority voting strategy where each classifier assigns the new instance one of two classes. The class with the majority votes is assigned to the instance. Crammer and Singer [7] pose the multi-class classification problem as a single optimization problem, rather than decomposing it into multiple binary classification problems. A comparison of the above approaches can be found here [8].
For a training set $(x_1, y_1)...(x_n, y_n)$ with labels $y_i \in \{1, \cdots, k\}$, the multi class problem can be posed as a constrained optimization problem with a quadratic objective function: min $\frac{1}{2}\sum_{i=1}^{k} \parallel w_i \parallel^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i$ s.t. $\forall y \leq k : [x_1 \cdot w_{yi}] \geq [x_1 \cdot w_y] + 100 * \Delta(y_1, y) - \xi_1$ $\cdots$ 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, $\Delta(y_j, y), 1 \leq j \leq n$ is the loss function that returns 0 if y_j equals y, and 1 otherwise and $\xi_i, 1 \leq i \leq n$ are the non negative slack variables which measure the degree of misclassification of the instance x_i. It must be noted that when the data is not linearly separable, the following kernel functions are used for classification: a) Linear Kernel: $K(x, y) = x^T y + c$ b) Polynomial Kernel : $K(x, y) = (\alpha x^T y + c)^d$ c) Radial Basis Kernel : $K(x, y) = exp(-||x - y||^2/2\sigma^2)$

6) **Information Retrieval Techniques:** The tokens identified by the model as "spell check","addition", "capitalization", and "punctuation" play an important role in trying to enhance search and retrieval on the archive. This is presented here for the completeness of the architecture, but described in detail in Section IV.

III. Empirical Evaluation

The experiments are performed on two machines, one of which is a linux HPC cluster and the other a dual core Mac machine with Intel Core i7 processor, 8GB RAM, 2.9 GHz of processor speed. In these experiments, the regularization parameter C is varied from 0.0001 to 10000 for the linear kernel. The performance of the algorithm is evaluated using 10-fold cross validation technique. The average loss on the test set and average CPU runtime are reported. Table IV presents the average loss on test set for different values of C ; AE_B, AE_P represents the average loss on the baseline dataset and proposed dataset respectively. AT_B, AT_P shows the average runtime (cpu sec) on the baseline dataset and proposed dataset respectively. We also experimented to build a model on the datasets using polynomial and radial basis kernel as shown in table **??**

C	AE_B	AE_P	AT_B	AT_P
.0001	99.43±.09	49.47±0.25	1.268±.06	0.11±.01
.1	99.43±.09	49.464±.47	2.512±0.01	0.061±0.01
10	99.43±.09	4.974±.5	2.512±0.01	0.17
1000	99.43±.09	1.743±.14	3.635±0.08	0.382±0.04
10000	99.43±.09	0	6.126±0.02	0.303±0.02

Table III
EXPERIMENT RESULTS USING LINEAR KERNEL

The code used to build the prototype system along with data generated are available from `https://github.com/megha89/`. It must be noted that the $SVM^{multiclass}$V2.2 package [9] was used for the implementation of Multi-class SVMs. For linear kernels, $SVM^{multiclass}$V2.20 is very fast and runtime scales linearly with the number of training examples. Training of non-linear kernels is very slow using the algorithm described in Section II.

C	AE_B	AE_P	AE_B	AE_P
0.0001	41.6±0.13	49.46±0.47	62.88±0.15	26.65±0.5
	AT_B	AT_P	AT_B	AT_P
0.0001	2217±81.7	574.9±14.6	1500.1±42.9	338.75±8.6

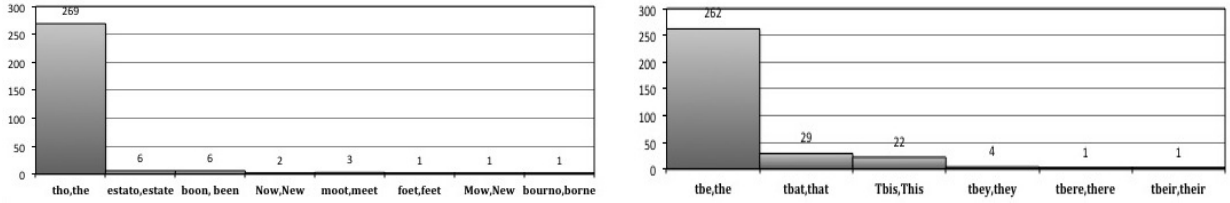
Table IV
EXPERIMENT RESULTS USING POLYNOMIAL AND RBF KERNEL

IV. Information Retrieval Techniques

The following details are relevant:

Query Set: To measure the retrieval effectiveness of the corrected text versus the garbled OCR, a list of 60 keywords for search was prepared by randomly sampling from the corrected words.

Software: PyLucene 3.6.2, a Python extension for accessing Java Apache Lucene was used as an IR software library for enabling full text indexing and search capabilites. Inverted indices are built for both the original and corrected corpus. Keywords from the query set are tested on both corpora and documents containing words in the query set are retrieved. The documents are ranked according to the frequency of the keyword present – higher the frequency, higher the rank.



(a) o changed to e

(b) b changed to h

Figure 3. Example of character level corrections

Evaluation: The main aim of this system is to improve the quality of text for use with an IR system. The metric used for our evaluation include the standard IR techniques of recall and precision. Recall is defined as the ratio of number of relevant documents returned to the total number of relevant documents for a query, in the collection. Precision is defined as the ratio of number of relevant documents returned to the total number of documents returned for a given query. The document to query relevance is manually determined by other computer science students. The table shows the precision at ten standard recall points for corrected OCR text over the raw OCR text.

Deployment: The full-scale deployment of this prototype system is being considered on old historic newspaper archives at California Digital Newspaper Collection and the New York Public Library among others, where user text corrections are used extensively for cleaning garbled OCR.

V. Related Work

Optical Character Recognition (OCR) is a commonly used method of digitizing printed texts so that it can be searched and displayed online, stored compactly and used in text mining applications. The text generated from OCR devices is often garbled due to variations in quality of the input paper, size and style of the font and column layout, its condition at the time of microfilming, choice of scanner, and the quality of the OCR software. Several techniques for post processing garbled OCR have been designed [10], [11], [12]. These include:

Dictionary based schemes: These algorithms use a dictionary to spellcheck misspelled OCR recognized words. They correct non-word errors - words that are recognized by the OCR device but do not correspond to any entry in the lexicon. Niwa et al. [13] proposed an OCR post error correction method based on pattern learning where a list of suitable candidates is generated from the lexicon and the best candidate is selected as a correction. Yannakoudakis and Fawthrop [14] conducted a study to create a set of rules based on common misspelling pattern and used them to correct errors. Cherkassky and Vassilas [15] use back propagation algorithms for correction.

Context based schemes: These algorithms perform error detection and correction based on the grammatical error and semantic context. They are able to correct real-word errors – words that are recognized by the OCR system and correspond to an entry in the lexicon. Tong and Evans [16] describe an automatic, context-sensitive, word-error correction system based on Statistical Language Modeling (SLM). The system exploits information from letter n-grams, character confusion probabilities and word bi-gram probabilities. Golding et al. [17] applies a part-of-speech (POS) tagger enhanced by word trigram model and a statistical Bayesian classifier to correct real-word errors in OCR text. Reynaert [18] presents a system for reducing the level of OCR-induced typographical variation in large text collections called Text-Induced Corpus Clean-up (TICCL). The system focuses on high-frequency words to be cleaned and gathers all typographical variants for any particular focus word that lie within the predefined Levenshtein distance. Bassil et al. [19], [20] propose a post-processing context-based error correction algorithm for detecting and correcting OCR non-word and real-word errors. The proposed algorithm is based on Google's online spelling suggestion. Abdulkader et al. [21] present a method for digitizing textual data by using neural network classifiers to estimate OCR errors, clustering similar errors, designing a user interface and using active learning to tune the error estimation from user labeled data. Velagapudi [22] uses a combination of classifiers – kNN classifier, multilayer perceptrons and SVM to discuss the effects of error correction on the classification accuracy of each method.

Very little work has been done on automatic *classification* of OCR error corrections[4] to get a clear understanding of the *nature* of errors encountered – Esakov et al. [23] suggest classification of output from the OCR engine as simple substitutions ($e \rightarrow c$), improper segmentation or multiple substitutions ($T \rightarrow' l, m \rightarrow rn, he \rightarrow b$), deletions and insertions (involving space) and unrecognized characters ($u \rightarrow \sim$). They use a new variant of the dynamic programming algorithm for classification. In [24], OCR documents are classified into fixed number of categories based on their content. The accuracy of their approach was best when evaluated using SVM among other algorithms. Daoason [25]

[4]Most prior work has dealt with evaluation of candidate corrections for OCR errors.

posits that OCR errors are not random – for example, it is extremely unlikely that the letter o will be misrecognized as x since they are very dissimilar. He classifies OCR errors as character errors (characters in the input document that are replaced with other characters), word errors (word in the input document is not correct or not present in the OCR generated text), or zoning errors (OCR software is unable to decipher zones correctly). Our work primarily focusses on *manual* error correction classification where the task of correcting OCR errors is outsourced to a crowd of workers.

Crowdsourced OCR correction Recruiting users and paying them small sums of money to tag and annotate text and images in large digital archives has become common practice (e.g. Amazon's Mechanical Turk, reCAPTCHA [26], ESP [27]) and Games with a Purpose [28]. A number of recent papers have evaluated the effectiveness of using Mechanical Turk to create annotated data for text and natural language processing applications [29]. In the same vein, many workshops and conferences have been organized on the theme of machine learning in human computation, crowdsourcing and collective intelligence[5].

Intuitively, the easiest way to correct garbled OCR text is to hire a group of people to edit it manually. Distributed Proofreaders (DP) [30] is a web-based project designed to facilitate proofreading of paper books into e-books and was meant to assist the project Gutenberg[6]. Wikipedia is yet another example of a large scale crowdsourcing project. Yamangil et al. [31] proposed a learning algorithm for mining wikipedia edit history using baseline Hidden Markov Model augmented with perceptron re-ranking. The model was trained on wikipedia edits and hence incorporated human corrections.

VI. 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.

In this paper, we present a system for Classification of Crowdsourced Text Correction which is capable of modelling user corrections using state-of-the-art machine learning techniques and retrieve categories which are likely to enhance search on the archive such as addition of words, elimination of typographical errors or addition of content. Information retrieval metrics are used to quantify the effectiveness of the user text correction. The prototype system is being considered for deployment on historic newspaper archives at the California Digital Newspaper Collection and the New York Public Library.

[5]See http://ir.ischool.utexas.edu/crowd/

[6]http://www.gutenberg.org/

Acknowledgment

This work is supported by funding from the National Endowment for Humanities, Grant No: NEH HD-51153-10. The authors would like to thank Stefan Boddie, DL Consulting, Ltd., New Zealand for sharing experiences with the Veridian software and Luis. C. Baquera, University of California, Riverside for providing data generated from the log files at CDNC.

References

[1] ADNP, "Australian newspapers digitization program," 2008. [Online]. Available: http://www.nla.gov.au/content/newspaper-digitisation-program

[2] R. Holley, "Crowdsourcing and social engagement: potential, power and freedom for libraries and users." november 2009.

[3] LDS, "Family search," 2005. [Online]. Available: https://familysearch.org/indexing/

[4] M. Ridge, "Frequently asked questions about crowdsourcing in cultural heritage," June 2011. [Online]. Available: http://openobjects.blogspot.co.uk/2012/06/frequently-asked-questions-about.html

[5] R. A. Wagner and M. J. Fischer, "The string-to-string correction problem," *J. ACM*, vol. 21, no. 1, pp. 168–173, Jan. 1974.

[6] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Support vector machine learning for interdependent and structured output spaces," in *Proceedings of the Twenty-first International Conference on Machine Learning*, ser. ICML '04, 2004, pp. 104–.

[7] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *J. Mach. Learn. Res.*, vol. 2, pp. 265–292, Mar. 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=944790.944813

[8] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *Trans. Neur. Netw.*, vol. 13, no. 2, pp. 415–425, Mar. 2002. [Online]. Available: http://dx.doi.org/10.1109/72.991427

[9] T. Joachims, "Multi-class support vector machine," Aug. 2008. [Online]. Available: http://www.cs.cornell.edu/people/tj/svm_light/svm_multiclass.html

[10] H. Fujisawa and C. L. Liu, "Classification and learning for character recognition: Comparison of methods and remaining problems in neural networks and learning in document analysis and recognition." *Machine Learning in Document Analysis and Recognition, Studies in Computational Intelligence*, vol. 90, pp. 139–161, 2008.

[11] J. Esakov, D. P. Lopresti, J. S. Sandberg, and J. Zhou, "Issues in automatic ocr error classification," *Proc. Third Annual Symposium on Document Analysis and Information Retrieval*, 1994.

[12] Y. Lecun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network." *NIPS*, 1989.

[13] H. Niwa, K. Kayashima, and Y. Shimeki, "Postprocessing for character recognition using keyword information," in *IAPR Workshop on Machine Vision Applications*, Dec 1992, pp. 519–522.

[14] E. J. Yannakoudakis and D. Fawthrop, "The rules of spelling errors," *Information Processing and Management*, vol. 19, pp. 87–99, 1983.

[15] V. Cherkassky and N. Vassilas, "Performance of back propagation networks for associative database retrieval," *In Proc. of the International Joint Conference on Neural Networks*, vol. 1, pp. 77–84, 1989.

[16] X. Tong and A. D. Evans, "A statistical approach to automatic ocr error correction in context," in *Proceedings of the Fourth Workshop on Very Large Corpora (WVLC-4*, 1996, p. 13.

[17] A. R. Golding and Y. Schabes, "Combining trigram-based and feature-based methods for context-sensitive spelling correction," in *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, ser. ACL '96. Stroudsburg, PA, USA: Association for Computational Linguistics, 1996, pp. 71–78. [Online]. Available: http://dx.doi.org/10.3115/981863.981873

[18] M. Reynaert, "Non-interactive OCR post-correction for giga-scale digitization projects," in *Computational Linguistics and Intelligent Text Processing*, 2008, vol. 4919, ch. 53, pp. 617–630.

[19] Y. Bassil and M. Alwani, "Ocr post-processing error correction algorithm using google online spelling suggestion," *CoRR*, vol. abs/1204.0191, 2012.

[20] ——, "Ocr context-sensitive error correction based on google web 1t 5-gram data set," *CoRR*, vol. abs/1204.0188, 2012.

[21] A. Abdulkader and M. R. Casey, "Low cost correction of ocr errors using learning in a multi-engine environment," in *Proceedings of the 2009 10th International Conference on Document Analysis and Recognition*, ser. ICDAR '09, 2009, pp. 576–580.

[22] P. Velagapudi, "Using hmms to boost accuracy in optical character recognition," 2005. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?rep=rep1&type=pdf&doi=10.1.1.141.7177

[23] J. Esakov, D. P. Lopresti, and J. S. Sandberg, "Classification and distribution of optical character recognition errors," *SPIE, Document Recognition*, vol. 2181, pp. 204–216, 1994.

[24] S. Laroum1, N. Bchet2, H. Hamza3, and M. Roche, "Hybred: An ocr document representation for classification task," *IJCSI International Journal of Computer Science Issues*, vol. 8, 2011.

[25] J. F. Daoason, *Post-correction of icelandic OCR text*. Faculty of Industrial Engineering, Mechanical Engineering and Computer Science, University of Iceland, 2012. [Online]. Available: http://skemman.is/stream/get/1946/12085/30520/1/Post-Correction_of_Icelandic_OCR_Text.pdf

[26] L. von Ahn, B. Maurer, C. Mcmillen, D. Abraham, and M. Blum., "reCAPTCHA: Human-based character recognition via web security Measures," Science, vol. 321, pp. 5895–, 2008.

[27] L. von Ahn and L. Dabbish., "Labeling images with a computer game." *In* Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '04*, pp. 319–326, 2004.

[28] L. von Ahn, "Designing games with a purpose," *Communications of the ACM*, vol. 51, 2008.

[29] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng, "Cheap and fast—but is it good: evaluating non-expert annotations for natural language tasks," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '08, 2008, pp. 254–263.

[30] C. Franks, "Distributed proofreaders," 2000. [Online]. Available: http://www.pgdp.net/c/

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