CCTC: Classification of Crowdsourced Text Correction

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

Megha Gupta
IIIT-Delhi, India.
meghag@iiitd.ac.in

Brian Geiger
University of California, Riverside
brian.geiger@ucr.edu

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

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 (ADNP 2008) enables 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 (Holley 2009). Picture Australia (NLA 2006) harvests digital images from other heritage institutions and encourages the public to upload their own images and tag them. FamilySearch (LDS 2005) 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. Galaxy Zoo (Community 20) is an online collaborative astronomy project in which users are invited to classify millions of galaxies from digital photos.

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 (Ridge 2011). In most citizen science projects, however, little statistical analysis is done on the user generated content. Thus 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) master and Web image files are generated; (3) metadata is assigned for each page to improve the search capability of the newspaper; (4) OCR software is run over high resolution images to create searchable full text and (5) OCR text, images, and metadata are imported into a digital library software program. 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 statistics generated from the correction process. The output from the system can be used to enhance search and retrieval on the archive.

Our 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.

[CHECK THIS] **Organization**: Section 2 describes the architecture of the proposed system. Section **??** describes the data obtained from crowd sourcing and pre-processing steps; Section **??** presents the design of the semi-automatic error

*The author is also a visiting assistant professor at IIIT-Delhi.

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

correction classifier and Section 3 presents empirical and scalability results using real-world data collected at CDNC. Section 4 discusses related work in OCR text correction with particular emphasis on systems requiring human intervention. Finally, Section 5 concludes the paper.

2 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 1(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 1(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 have been chosen for our work in this project. Each log file is generated at the issue-level and contains XML data about the pages in the issue. Table 1 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.

<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

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 44,023 tokens of which 21,113 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 (Wagner and Fischer 1974) 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) *Same Length* : 1, if both the old word and new word have same length and 0 otherwise. For example, the feature is 1 for the tokens "Spll," and "Union" as both the tokens have same number of characters.
     (b) *Edit Distance Zero*: 1, if both the words are exactly same and 0 otherwise. For example, the feature is 1 for tokens "Under" and "Under".
     (c) *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.
     (d) *Edit Distance Below Three*: 1, if less than three edit operations are required to convert old word to new word and 0 otherwise. For "Stales" and "Stables", value is 1 as less than three edit operations are required to convert from old to new token.
     (e) *Edit Distance is 1 and Case 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 "Stales" and "Stables", the value is 0 as there are two conditions to be met. One, when both the tokens have edit distance exactly 1 and second, when there is a upper to lower or lower to upper case change in the first character. Here, the edit distance is exactly 1 but 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 (!"#$%&'*+,-./:;<=>?@[\]^_`{|}~) . For "Spll," and "Union", the value is 1 as the old and new text differ in one of the following punctuation marks
   - **Label Constructor:** The errors rectified by the users are categorized as Spellcheck Error, Addition of a new word, Capitalization Error, Typo and Punctuation error. The logic used in generation of the features can be summarized by the flowchart depicted in Figure **??**. Specifically,
     (a) *Class 1-Spellcheck error:* When the edit distance is between 1 and 3. For example, "mounten" and "mountain".
     (b) *Class 2-Addition of a new word:* When the edit distance is more than 3. For example, "at" and "attend".
     (c) *Class 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".
     (d) *Class 4-Typo:* When the edit distance is exactly one and case change is 0. For example, "teh" and "the"
     (e) *Class 5-Punctuation error:* When the two strings differ by special characters contained in the set

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

(!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~). For example, "residents" and residents

(f) *Class 6-No correction:* When the old and new text are same. For example, plant and plant

The distribution of the these classes in the dataset is shown in Table 2.

Class	No. of Instances
Spellcheck	1970
Addition of new word	8732
Capitalization	261
Typo	2572
Punctuation	6602
No correction	23885
Total	44022

Table 2: Class Distribution

4. **Model Construction:** The crowdsourced text correction model is built using a Multiclass Support Vector Machine algorithm (Joachims 2008). For a training set $(x_1, y_1)...(x_n, y_n)$ with labels $y_i \in \{1, \cdots, k\}$, the multi class problem is posed as a constrained optimization problem with a quadratic objective function which can be stated as:

$$\min \frac{1}{2} \sum_{i=1}^{k} \parallel w_i \parallel^2 + \frac{C}{n} \sum_{i=1}^{n} \xi_i \quad (1)$$

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

. . . . . .

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

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 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. **Information Retrieval Techniques:** The tokens identified by the model as "spell check","addition of a new word", and "Typo" play an important role in trying to enhance search and retrieval on the archive. The following details are relevant:

Query Set: To measure the retrieval effectiveness of the corrected text versus the garbled OCR, a list of 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 Index was built on each corpus with the fields stored as file name, file path and file contents making retrieval faster. Experiments were run on the indexes created on both the datasets using the above mentioned query set. Only documents containing words in the query set were retrieved. The documents are ranked according to the frequency of the keyword present in the document. Higher the frequency, higher the ranking order.

Evaluation: The metric we used to compare the ranked retrieved documents is Spearman's rank correlation coefficient(ρ).

Let y_i be the rank ordering of the documents retrieved from the corrected corpus and x_i the rank ordering for the garbled text. Let d_i = x_i - y_i. Then Spearman's rank correlation ρ is given by:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)} \quad (2)$$

Example: Query word "Jackson", the evaluation of ρ (Worst Case) is shown in Table 5. Here, $\sum d_i^2$ = 11 and ρ = 0.45.

Y_i	X_i	y_i	x_i	d_i	d_i^2
12-01-1900	21-10-1910	1	10(4)	-3	9
09-12-1910	09-12-1910	2	2(1)	0	0
08-06-1900	23-09-1910	3	7(2)	1	1
01-04-1904	02-12-1910	4	8(3)	1	1
25-05-1900	23-12-1910	5	11(5)	0	0

3 Empirical Evaluation

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.

4 Related Work

Optical Character Recognition

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 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. These include:

Dictionary based schemes: They use a lookup dictionary to spellcheck misspelled OCR recognized words. They correct non word errors, a word that is recognised by the OCR device but does not correspond to any entry in the lexicon. (Niwa, Kayashima, and Shimeki 1992) 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. (Hull 1996) proposed using HMM to integrate syntactic information into the post-processing error correction. The suggested model achieved a higher rate of error correction due to its statistical nature in selecting the most probable candidate for a particular misspelled word.

Context based schemes: They 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 OCR system and does correspond to an entry in the lexicon.(Golding and Schabes 1996) 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.

Manual error correction schemes: Our work primarily focusses on manual error correction schemes where the task of correcting OCR errors is outsourced to a crowd of workers. By engaging a large crowd of people for ideas, skills or participation the quality of work done is superior and relatively inexpensive.

Crowdsourced OCR correction

(Brabham 2008) was the first to define "crowdsourcing" as an online, distributed problem-solving and production model. The most comprehensive encyclopedia this world has ever seen is Wikipedia which is a famous example of a crowdsourced project.
(Yamangil and Nelken 2008) proposed a learning algorithm for mining wikipedia edit history using baseline Hidden Markov Model augmented with perceptron reranking. As the model was trained on wikipedia edits, its attuned to human corrections. Other initiatives that include human intervention is (Abdulkader and Casey 2009). This work gives a low cost approach for digitizing textual data by using methods like neural network classifier to estimate OCR errors, clustering similar errors, designing a user interface and using active learning to tune the error estimation by using user labeled data. An important work (Velagapudi 2005) that uses a combination of classifiers, a kNN classifier, a multilayer perceptron and SVM discusses the the effects of error correction on the classification accuracy of each method. In another work (Laroum1 et al. 2011), instead of error classification OCR documents were classified into fixed number of categories based on their content. The accuracy of their approach was best when evaluated using SVM among other algorithms.

Multi-class SVM

Among the different methods used for solving SVM multi classification problem, some of the relevant works in related areas. a) *one-versus-all (OVA classification)* uses 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* in its training phase is the same as the one-against-one method by solving k(k-1)/2 binary SVMs. However, in the testing phase, it uses a rooted binary directed acyclic graph which has k(k-1)/2 internal nodes and k leaves. This is superior to other multiclass SVM algorithms in both training and testing time. d)*error correcting output codes*, (Wang et al. 2010) proposed an ECOC based probabilistic approach to solve multi class SVM problem. In the training phase a coding scheme is predefined, and a special model is trained by samples. In the classification stage, besides the labels from SVM, posterior probabilities of labels are also calculated. They are used to compute probability estimates of categories. 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. A comparison of the above approaches can be referred here (Hsu and Lin 2002).

5 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 3 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

Abdulkader, A., and Casey, M. R. 2009. 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*, ICDAR '09, 576–580. IEEE Computer Society.

ADNP. 2008. Australian newspapers digitization program.

Ahn, L. v.; Maurer, B.; McMillen, C.; Abraham, D.; and Blum, M. 2009. recaptcha, digitizing books one word at a time.

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

books, G. 2013.

Brabham, D. C. 2008. Crowdsourcing as a model for problem solving: An introduction and cases. *Convergence: The International Journal of Research into New Media Technologies* 14.

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.

Community, A. 20. Galaxy zoo.

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

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

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. Association for Computational Linguistics.

Golding, A. R., and Schabes, Y. 1996. Combining trigram-based and feature-based methods for context-sensitive spelling correction. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, ACL '96, 71–78. Stroudsburg, PA, USA: Association for Computational Linguistics.

Hoi, S. C. H.; 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*, WWW '06, 633–642. New York, NY, USA: ACM.

Holley, R. 2009. Crowdsourcing and social engagement: potential, power and freedom for libraries and users.

Hsu, C.-W., and Lin, C.-J. 2002. A comparison of methods for multiclass support vector machines. *Trans. Neur. Netw.* 13(2):415–425.

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.

Hull, J. J. 1996. Incorporating language syntax in visual text recognition with a statistical model. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 18, 1251–1256.

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. Making large-scale support vector machine learning practical. In *Advances in kernel methods*. MIT Press. 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.

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

LDS. 2005. Family search.

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

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

Niwa, H.; Kayashima, K.; and Shimeki, Y. 1992. Postprocessing for character recognition using keyword information. In *IAPR Workshop on Machine Vision Applications*, 519–522.

NLA. 2006. Picture australia.

Platt, J. C.; Cristianini, N.; and Shawe-taylor, J. 2000. Large margin dags for multiclass classification. In *Advances in Neural Information Processing Systems*, 547–553. MIT Press.

Project, G. B. L. 2013.

Rajaraman, A.; Leskovec, J.; and Ullman, J. D. 2013. *Mining Of Massive Datasets*. Cambridge University Press.

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*. Springer. chapter 53, 617–630.

Ridge, M. 2011. Frequently asked questions about crowdsourcing in cultural heritage.

Tanner, S.; Muoz, T.; and Ros, P. H. 2009. Measuring mass text digitization quality and usefulness: Lessons learned from assessing the ocr accuracy of the british library's 19th century online newspaper archive. *D-Lib Magazine* 15.

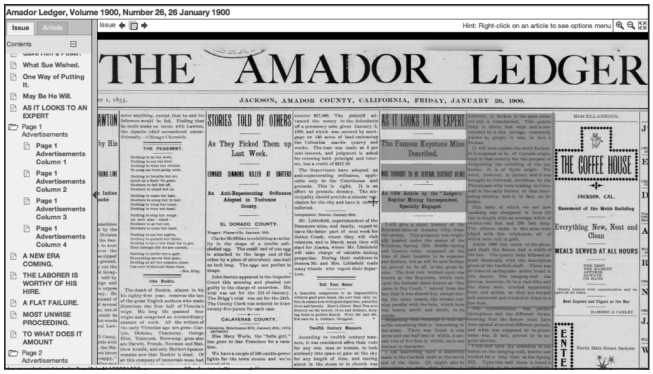
Tong, X., and Evans, A. D. 1996. A statistical approach to automatic ocr error correction in context. In *Proceedings of the Fourth Workshop on Very Large Corpora (WVLC-4*, 13.

Velagapudi, P. 2005. Using hmms to boost accuracy in optical character recognition.

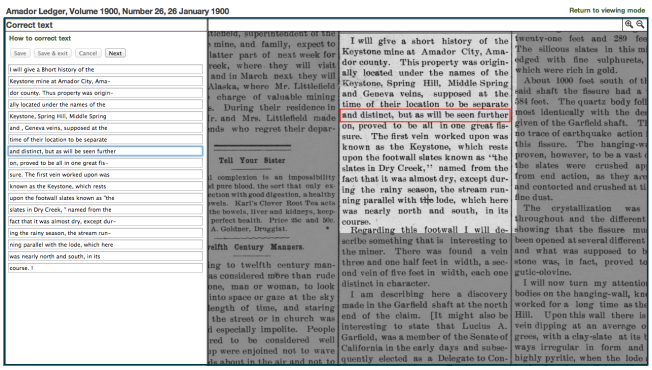
Wagner, R. A., and Fischer, M. J. 1974. The string-to-string correction problem. *J. ACM* 21(1):168–173.

Wang, Z.; Xu, W.; Hu, J.; and Guo, J. 2010. A multiclass svm method via probabilistic error-correcting output codes. 1–4.

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



(a) Scanned newspaper "The Amador Ledger" highlighting an article to be corrected by a user.



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