Classification of Crowdsourced Text Correction

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

Haimonti Dutta[*]
Department of Management Science and Systems
State University of New York, Buffalo
New York, 14260
haimonti@buffalo.edu

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.This prototype system is expected to be deployed on historical newspaper archives that make extensive use of user text corrections.

1. INTRODUCTION
In this paper, we focus on understanding the nature of text corrections done by users of an old historic newspaper archive. 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.

[*]The author is also affiliated to the Institute of Data Science and Engineering (IDSE), Columbia University and is an adjunct professor at IIIT-Delhi.
[1]http://cdnc.ucr.edu/cgi-bin/cdnc

2. SYSTEM COMPONENTS
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.

2. **Log Files:** All corrections performed by the annotators are recorded in log files. 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 new text from the log file is tokenized by white-spaces resulting in total 44,022 tokens of which 21,108 are corrected by the annotators.
   - **Feature Constructor: Word-level feature construction:** (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. (d) **Edit Distance Above One**: 1, if more than one edit operation is required to convert old word to new word and 0 otherwise. (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

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

upper case to lower case or vice versa. (f) **Punctuation Difference**: 1, if both the old text and new text differ in any of the following punctuation marks (!”#$%&’*+,-./:;<=>?@[\]≏‘{|}˜).

Character-level feature construction: The tokenization process of the text in the log files was modified to enable the addition and deletion classification of text using a character. The corrected tokens were split into character-level data contributing 58,963 instances. Of 58,963 instances, we encountered that 55,612 had same character-level corrections indicating that the OCR makes the same type of mistakes repeatedly. The maximum number of redundant corrections noted were deletion of punctuations and special characters. In order to test the performance of the manually crafted (word-level) features of the dataset, we used association rule mining algorithm to generate rules. Hence, the character-level feature construction acts as a baseline method. The number of features and instances are 6631 and 3351 respectively in the character-level data set.

- **Label Constructor:** (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. (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. (c) **Punctuation:** When the difference in the length of strings is non-zero and they differ by special characters contained in the set (!”#$%&’()*+,-./:;<=>?@[\]≏‘{|}˜). (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.(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.

  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].

4. **Model Construction:** The model for classifying crowdsourced text correction is built using a Multiclass Support Vector Machine algorithm [1]. 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.

5. **Information Retrieval Techniques:** The main aim of this system is to improve the quality of text for use with an IR system. The metric used for evaluation here is mean reciprocal rank (MRR) which produces a list of possible ranked responses to a sample of queries, Q. The reciprocal rank is the multiplicative inverse of the rank of the first correct answer. The first correct answer is decided by the users. The average of the reciprocal ranks over a sample of queries is the mean reciprocal rank.

[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

3. RESULTS

The performance of the algorithm is evaluated using 10-fold cross validation technique. The **A**verage loss **E**rror (AE) and **A**verage running **T**ime (AT) from cross validation for baseline (character-level) and proposed (word-level) dataset respectively are represented as AE_b, AE_p. Table 1 shows the result of the experiment, error and running time (cpu seconds) using linear kernel for different values of C. Table 2 shows the result of the experiment, error and running time (cpu minutes) using polynomial and radial basis kernel. The multi-class SVM algorithm using polynomial kernel gives the best results in both the datasets, word-level and character-level. In case of automatically generated character-level dataset, the performance of the algorithm does not improve by varying the regularization constant in contrast to the performance of the manually crafted word-level dataset. In case of word-level dataset, as the value of regularization constant increases, the average loss of the learning model decreases. However, the time taken to train the model is considerably high. It must also be noted that one can learn less accurate but faster models with linear or RBF kernels. Thus, it may be useful to consider a trade-off between accuracy of learnt models versus time taken for classification in large scale deployments.

4. CONCLUSIONS

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. We also compared the two datasets, word-level and character-level by applying the machine learning algorithms on each. The manually crafted word-level dataset performed better in terms of average loss error. However, the running time of the algorithm was considerably high. Thus, there is a need to improve the algorithms in order to make them compatible with large scale datasets.
Information retrieval metric, mean reciprocal rank is used to quantify the effectiveness of the user text correction. The higher value of the MRR in the case of corrected corpus indicates that the relevant results are ranked higher in the retrieved set of documents.

5. REFERENCES

[1] 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*, ICML ’04, pages 104–, 2004.

Table 1: Results using linear kernel

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 2: Results using polynomial and rbf kernels

	Polynomial		RBF	
C	AE_b	AE_p	AE_b	AE_p
.0001	42±.13	50±.47	63±.15	27±.5
100	42±.13	.33±.2	63±.16	3±.4
1000	42±.13	0±0	63±.16	1.7±.2
C	AT_b	AT_p	AT_b	AT_p
.0001	37±1.4	10±0.2	25±0.7	6±0.2
100	326±11.8	1239±118	540 ± 110	404±34
1000	942±17.4	764±93	537±111.72	957±184