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). 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.

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**[1]
2. **Log Files:** All corrections performed by the annotators are recorded in log files. The following information is provided about the corrections: (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* (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: The entire word/token modified by the patrons is used for assigning the binary values to the hand crafted features. (a) **Difference Length Zero** : 1, if both the old and new tokens have same length. (b) **Difference Length Above One** : 1, if the length of new token exceeds the length of old. (c) **Edit Distance One**: 1, if single edit operation can convert old to new token. (d) **Edit Distance Above One**: 1, if multiple edit operations convert old to new token. (e) **Edit Distance is 1 and Case Change**: 1, if the first character of old/new token is the upper/lower case of new/old token provided edit distance is 1. (f) **Punctuation Difference**: 1, if the old and new token differ in any of the punctuation marks.
     Character-level feature construction: In order to test the performance of the hand crafted features, we generated automatic rules using association rule mining algorithm.

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

Here, only those characters of the tokens which were modified by the patrons are used to assign binary values to the automatically generated features. We noticed that out of 58963 instances of modified characters, 55612 had same corrections indicating that OCR makes same type of mistakes (mostly punctuations and special chars). The character-level feature construction acts as a baseline method to compare and analyse the error classification against word-level feature construction.

- **Label Constructor:** The labels are used to categorise errors depending on the factors like, length, edit distance, punctuations, lower/upper case characters in the tokens. This categorisation of errors can help us understand the kinds of errors an OCR device commits. This can help us in improving the areas which contribute to the maximum errors thereby improving the overall efficiency of the device. (a) **Addition:** For example, "6RAVR" and "GRAVEL". (b) **Deletion:** For example, "VVe" and "We". (c) **Punctuation:** For example, "Ladies!" and "Ladies". (d) **Capitalization:** For example, "largest" and "Largest". (e) **Spellcheck:** For example, "hanger" and "banger". It must be noted that by design tokens are always assigned to one class based on priority, although in principle it may be possible to assign them to multiple classes[2].

4. **Model Construction:** The model for classifying crowdsourced text correction is built using a Multiclass Support Vector Machine algorithm[3]. 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:** A set of queries were applied on both the datasets, that is raw OCR and corrected OCR individually. For each of the queries, we chose the most relevant document from the retrieved results and observed the rank of the most relevant document in both the set of results. The MRR for the raw OCR data was noted to be 0.355 whereas for corrected data it was 0.65415. The higher value of MRR denotes that relevant results get higher priority in corrected corpus.

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. Thus, 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.

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

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 that help us to know the types of errors mostly committed by the OCR devices. This can be step towards improving the performance of the devices so that the goal to enhance search on the archive can be achieved. 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 which calls for the need to improve the algorithms in order to make them compatible with large scale datasets.

5. ACKNOWLEDGMENTS

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.

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