Estimating Residual Error Rate in Recognized Handwritten Documents Using Artificial Error Injection

Edward Lank, Ryan Stedman, and Michael Terry
David R. Cheriton School of Computer Science
University of Waterloo
200 University Ave. W., Waterlo, ON, Canada, N2L 3G1
<lank, rstedman, mterry>@cs.uwaterloo.ca

ABSTRACT
Both handwriting recognition systems and their users are error prone. Handwriting recognizers make recognition errors, and users may miss those errors when verifying output. As a result, it is common for recognized documents to contain *residual errors*. Unfortunately, in some application domains (e.g. health informatics), tolerance for residual errors in recognized handwriting may be very low, and a desire might exist to maximize user accuracy during verification. In this paper, we present a technique that allows us to measure the performance of a user verifying recognizer output. We inject artificial errors into a set of recognized handwritten forms and show that the rate of injected errors and recognition errors caught is highly correlated in real time. Systems supporting user verification can make use of this measure of user accuracy in a variety of ways. For example, they can force users to slow down or can highlight injected errors that were missed, thus encouraging users to take more care.

Author Keywords
Handwriting recognition, residual error, artificial error.

ACM Classification Keywords
H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

General Terms
Measurement, Performance.

INTRODUCTION

Despite increases in accuracy, recognition errors are still present in the output of handwriting recognition systems. Many of these errors can be traced to variations in handwriting between individuals and to the use of dictionaries of recognizable words to identify target classes for recognizer output [5]. If a student is recognizing their classroom notes, the presence of handwriting recognition errors might be of only minor concern. However, for handwriting recognition to be usable in applications where a high output accuracy is needed, such as health care settings, user verification of recognizer output is necessary.

If handwriting is performed on a Tablet PC and the recognizer fires in real time, error discovery and correction are quite simple. The amount of recognizer output being validated is small, and a user can simply look to compare two short text strings (one of which they just entered) for correspondence. However, users frequently need to validate a larger amount of recognizer output: sometimes many pages of text or many data forms. Beyond the amount of data recognized, users of verification interfaces may need to validate data that was written by other writers. In these cases, the verification task is further confounded by a lack of context; the verifier frequently cannot remember what was written, either because of the volume or because they were not the author of the data being recognized and verified.

The premise of this work is that both recognizers and humans are error prone. Recognizers will occasionally misrecognize input, resulting in recognition errors. Users are also likely to miss some of the recognition errors when verifying recognizer output. As a result, even after handwriting recognition has been verified by a human reader, *residual errors* [1, 2, 3] may still exist in the output. Residual errors are particularly likely when verification is done on a large amount of recognizer output, e.g. when recognition results for tens or hundreds of documents are verified. Furthermore, there exist domains where residual errors are costly; one example would be the use of recognizers on data collected in health care settings.

In this note, we demonstrate a technique for *automatically determining user performance*, in real time, when verifying handwriting recognizer output. To determine user performance, we inject artificial recognition errors into the handwriting recognizer output. As a result, each document, or a subset of the documents, validated by a user has both recognition errors from the handwriting recognizer (recognizer errors) and artificial errors which we inject (injected errors).

We find that the percentage of injected errors caught strongly correlates with the percentage of recognition errors caught by users verifying recognizer output. The correlation holds over a window of time within the task, allowing us to use the number of injected errors found by the user at any point in time as a measure of a user's real-time accuracy in verifying recognizer output.

How can systems make use of these estimates? Systems could reject output if users are too inaccurate in verification. As well, if techniques exist to enforce accuracy on users – for example by encouraging them to slow down, or by highlighting the injected errors they missed in a document – the system could use these technique to ensure documents are sufficiently free from residual errors.

In the remainder of this note, we highlight related work in using simulated errors to test user performance, we outline our technique for injecting errors to measure performance in a verification task, and we describe an experiment that validates our technique.

RELATED WORK

Artificially injecting errors to test performance has been used in other domains. For example, fault injection [4] is a software engineering technique that measures tolerance of software to program bugs. By injecting faults into a software system, testers can verify that error handling code functions correctly. As well, in optical character recognition systems (OCR), simulated errors have been used to assess how accurate recognizers must be to justify their use [1]. By varying error rate, researchers were able to show that, if recognition rates were less than 94% it was faster to type the document, and if recognition rates were less than 98% the residual error rate was worse with OCR systems.

TECHNIQUE

When working with ground-truthed data sets, measuring user performance during verification is trivial. Given recognizer errors, get users to verify the output, then count how many errors a user misses. While this works well when ground-truth exists for the data, it is not useful in real world recognition tasks where ground truth does not exist. The system has no knowledge of the number of errors the recognizer makes, and the number of recognizer errors may vary widely over a set of inputs being recognized.

Our technique works as follows. When a system wishes to measure a user's performance on the verification task, it injects artificial errors into the handwriting recognition results, over-writing presumably correctly recognized words or numbers. Users verifying recognition results will correct both recognition errors and injected errors. We hypothesize that the rate of correction of recognition errors and injected errors will be similar.

The challenge with error injection is the need to ensure the visibility of the artificial errors is equivalent to that of real errors. If the artificial errors are more or less apparent than real recognition errors, the chance of a user discovering artificial errors may differ from their chances of identifying an artificial error.

Creating Realistic Artificial Errors

In our experiment, we use the Microsoft handwriting recognizer bundled with Windows Vista as a test bed for our technique. We choose this recognizer because it has relatively high accuracy and because it is widely available, making it a *de facto* standard.

The best resource to create a false error for a recognized result is the n-best list from the recognizer for that result. Any result from the n-best list of a recognizer is a result of similarities in word shape sequence: ascenders, descenders, peaks and valleys occur in similar sequence to the original handwritten input. N-best list results therefore approximate the shape of the handwritten text. However, in using results from the n-best list, another possible problem occurs: What if the false error selected is the correct recognition result? Injecting a correct result causes two problems. The user will not flag the correct result, and the system will not know whether it is best to use the original recognizer result or the injected result as the final interpretation for the input.

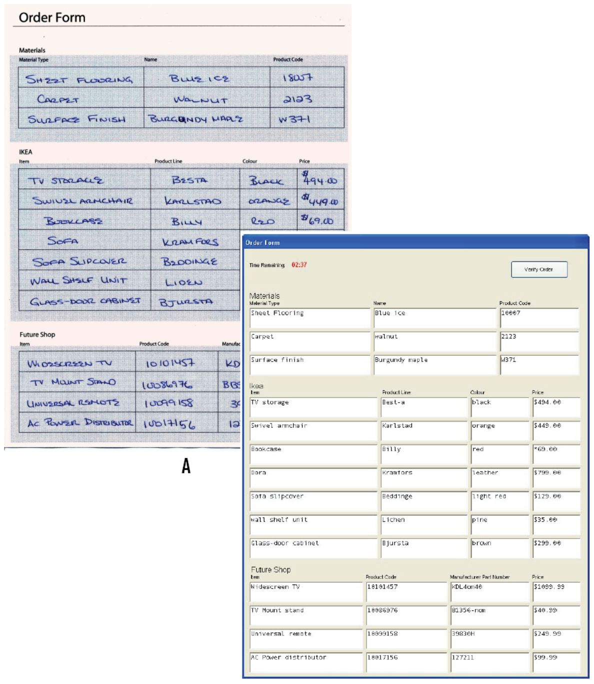
An analysis of recognition results from the Microsoft handwriting recognizer supports this concern. Specifically, of an analysis of several hundred real recognition errors from the recognizer, 35% of these recognition errors had the correct result present in the n-best list. A more detailed analysis of the n-best list, however, presents a potential solution. Of those 35% of errors with correct results in the n-best list, 13% had the correct result as the first alternate, 10% as the second alternate, and 4% as the third alternate. The correct result was 4th or lower only 8% of the time. To select a candidate for injection, we skip the first three alternates in the n-best list, and select randomly from the rest of the list (4th through 9th alternate).

A related issue that arises in selecting errors from the n-best list is capitalization variants. For the Microsoft recognizer, it is common for the n-best list to contain capitalization variants of the top candidate. To address this, any result from the n-best list was not a candidate for injection if it only differed based on capitalization from the recognition result. In this case, another candidate in the list was used.

Despite the intricacies we use when selecting from the n-best list, in practice we find there exists a 1% - 2% chance that we inject the correct recognition result when we desire to inject an error. In a real-world implementation of our technique, we could further address the problem by bringing to a user's attention any injected errors that they missed. Users could then indicate whether the injected error is the correct recognition of the ink.

EXPERIMENT

During previous research, 400 data forms were filled out using an Anoto pen, as shown in Figure 1a. The on-line ink from the Anoto pen was downloaded and converted to timestamped Microsoft ink. Handwriting recognition was then performed on the ink using the Microsoft recognizer.



A

Order Form

Materials

Material Type	Name	Product Code
SHEET FLOORING	BLUE ICE	18007
CARPET	WALNUT	2123
SURFACE FINISH	BURGUNDY MAPLE	W371

IKEA

Item	Product Line	Colour	Price
TV STORAGE	BESTA	BLACK	$494.00
SWIVEL ARMCHAIR	KARLSTAD	ORANGE	$449.00
BOOKCASE	BILLY	RED	$69.00
SOFA	KRAMFORS		
SOFA SLIPCOVER	BEDDINGE		
WALL SHELF UNIT	LIDEN		
GLASS-DOOR CABINET	BJURSTA		

Future Shop

Item	Product Code	Manufac
WIDESCREEN TV	10101457	KD
TV MOUNT STAND	10086976	BB
UNIVERSAL REMOTE	10099158	3
AC POWER DISTRIBUTOR	10017156	12

B

Order Form

Time Remaining 02:37

Verify Order

Materials

Material Type	Name	Product Code
Sheet Flooring	Blue ice	10007
Carpet	walnut	2123
Surface finish	Burgundy maple	W371

Ikea

Item	Product Line	Colour	Price
TV storage	Best-a	black	$494.00
Swivel armchair	Karlstad	orange	$449.00
Bookcase	Billy	red	*69.00
Dora	Kramfors	leather	$799.00
Sofa slipcover	Beddinge	light red	$129.00
wall shelf unit	Lichen	pine	$35.00
Glass-door cabinet	Bjursta	brown	$299.00

Future Shop

Item	Product Code	Manufacturer Part Number	Price
Widescreen TV	18101457	KDL4om40	$1099.99
TV Mount stand	18086976	B1356-nom	$40.99
Universal remote	18099158	39830H	$249.99
AC Power distributor	18017156	127211	$99.99

Figure 1: A) Handwritten data forms; and B) Interface displaying recognition results.

Using this existing data set, legible data forms were independently selected by two researchers, and a common subset of 20 legible forms were used for the experimental task. We selected legible forms to avoid the problem that arises when a user verifying recognizer output cannot decipher the original handwriting, making verification particularly difficult. The Microsoft recognizer was used to interpret the handwriting, and the results were displayed in an interface on a desktop monitor with 1280X1024 pixel resolution. The layout of the screen ensured that scrolling was not necessary, as shown in Figure 1b. As is typical in editing tasks, clicking once in each field inserted the cursor, clicking twice selected the word, and clicking three times selected all data in the field.

The Microsoft recognizer's error rate was 23% (mean errors per form = 11.3, SD = 5.2, min = 5, max = 18), despite biasing the recognizer for field data types on the form. It is not unusual to see high error rates on real world input; handwriting recognizers rarely perform as well on real world data as they do in controlled tests. The large range in recognition errors from form to form is also not unusual.

In each recognized form, ten artificial errors were randomly injected into the results. To avoid the possibility of grouping injected errors, the ten errors were spread out proportionally among the three different blocks of each form (shown in Figure 1a). Two errors were randomly injected into the first block, five in the second, and three in the third.

Method and Participants

12 participants (7 female) were recruited from the university campus. Participants' ages ranged from 18 – 38 (mean = 22.5, sd = 5.4) and they were paid $15.

In any experimental evaluation of verification tasks, there is a need to simulate realistic pressures. We wanted to encourage the participants to be as fast and as accurate as possible. It was, however, unclear to us whether time pressure or an incentive would most effectively motivate participants. As a result, we divided our participants into two groups: The first had a 3 minute time limit on the verification of each form; the second was told that the top two performing participants (fastest and most accurate) would receive a $15 gift certificate at the end of the study.

RESULTS

The results of the experiment are summarized in Table 1. Timed participants are listed as T1 – T5. Incentive participants are listed as I1 – I6. An error occurred during data collection in T6's experimental session, causing data from that session to be invalid. T6's data was, therefore, excluded from analysis. T6's partial data set had no discernable effect on the results reported below.

Participant	Average Injected Errors Caught	Average Recognition Errors Caught
T1	0.951	0.965
T2	0.792	0.825
T3	0.607	0.625
T4	0.941	0.925
T5	0.791	0.82
I1	0.96	0.916
I2	0.967	0.935
I3	0.967	0.925
I4	0.97	0.96
I5	0.945	0.885
I6	0.961	0.98

Table 1: Per participant performance for injected and recognition errors.

Overall correlation between injected and recognition errors caught during the verification task was extremely high for our timed participants (r^2 = 0.99, p = 0.001) and for all participants (r^2 = 0.97, $p < 0.001$). For our incentive participants, the desire to value accuracy over speed resulted in performance skewed toward accuracy. As a result, the high number of errors caught and the narrow range results in poor correlation (r^2 = 0.66, p = 0.15). As well, participant I6 is an outlier (based on an analysis of user-effects in SPSS). Omitting this user changes correlation to r^2 = 0.93, p = 0.022. However, as our overall correlation (r^2 = 0.97) is quite high and statistically significant, we perform further analysis on all participants.

Human verifiers perform better or worse based on individual variation (some take more care than others).

Also, during any verification task, habituation or fatigue may cause the residual error rate to vary. One open question is whether our technique can measure performance in real-time, for example on a form-by-form basis, or over a window of a small set of forms. To answer this question, we looked at correlation on individual forms and at correlation over a window of four forms.

At the level of individual forms (n = 220 data points), we also see a statistically significant correlation of $r^2 = 0.59$ ($p < 0.001$). While statistically significant, the magnitude of the correlation coefficient may be a cause for concern. However, because there are only 10 injected errors and between 5 and 18 recognition errors on each form, this correlation has significant noise.

To increase the number of injected errors and smooth our data, we consider a sliding window of four forms, or 40 injected errors. Given this data set (n = 187 data points, or 17 data points per user, see Figure 2), we see a highly significant correlation, $r^2 = 0.84$ ($p < 0.001$). Given the high correlation coefficient, we can conclude that percentage of injected errors caught by a user is a strong and reliable predictor of the percentage of recognition errors caught.

GENERALIZING RESULTS TO REAL WORLD APPLICATIONS

There are two potential shortcomings of error injection. The first is a practical limitation: by injecting errors, we are making the verification task more time consuming. In our case, we almost double the workload of the users verifying recognizer output. The second is an experimental concern: by injecting errors into every form, the higher perceived error rate may invalidate the correlations we observed. We deal with each of these objections in this section.

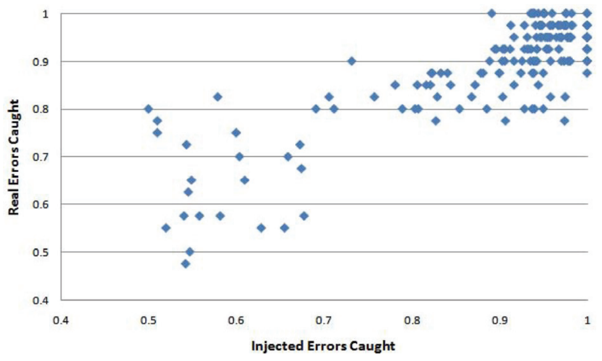


Figure 2: Injected vs. recognition errors caught over a 4-form window.

In our evaluation, every form verified by a user had both injected and recognition errors. Clearly, in a real world system, injecting errors so frequently would not be cost effective due to the increase in the workload of users verifying output. However, errors need not be injected so frequently. Systems could inject errors at certain pre-defined intervals to measure user performance. The standard deviation of the recognizer's error rate is already significant on a form-by-form basis, so this might be transparent to the user. As another options, rather than having both injected and recognition errors on any forms, systems could inject errors into forms rather the user had previously corrected (even making use of the original recognition errors identified) and measure user accuracy when repeating the verification of the previously corrected form. This would limit the increased workload on any one form, as the form would only have injected or recognition errors, not both. The increased workload would then be a function of how frequently user performance is measured.

Next, we note that injecting errors in every form boosts the perceived error rate of the recognizer significantly. However, the number of recognition errors per form also varied significantly from 5 to 18. With injected errors, errors varied from 15 to 28 on each form. While this may seem like a large number of errors per form, the high variability of this number and the good correlation on a form-by-form basis give us confidence that our results generalize across a significant range of recognition rates.

CONCLUSION

We present a technique that accurately estimates user performance when users are verifying handwriting recognition results. The technique uses artificially injected errors, and we demonstrate high correlation between percentages of injected and real recognition errors caught. Having accurate estimates of user performance can allow systems to enforce needed accuracy levels, increasing user confidence in recognized data sets.

ACKNOWLEDGEMENTS

Funding for this research was provided by the Natural Science and Engineering Research Council of Canada, NSERC. Experimental design was approved by the Office of Research Ethics, University of Waterloo.

REFERENCES

1. Cushman, W. H., Ojha, P. S., and Daniels, C. M. "Usable OCR: what are the minimum performance requirements?" In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* CHI '90, pp. 145-152.
2. Mankoff J., and Abowd, G. D. "Error Correction Techniques for Handwriting, Speech, and Other Ambiguous or Error Prone Systems." GVU Technical Report: GIT-GVU-99-18, 1999.
3. Mankoff, J., Hudson, S. E., and Abowd, G. D. "Providing integrated toolkit-level support for ambiguity in recognition-based interfaces." In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* CHI '00, pp. 368-375.
4. H. D. Mills, "On the Statistical Validation of Computer Programs," Technical report FSC-72-6015, IBM Federal Systems Division 1972.
5. Plamondon, R. and Srihari, S. N. "Online and off-line handwriting recognition: A comprehensive survey," in *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 22:1, pp. 63 – 84.