Correcting Noisy OCR: Context beats Confusion

John Evershed
Project Computing
Canberra
Australia
John.Evershed@projectcomputing.com

Kent Fitch
Project Computing
Canberra
Australia
Kent.Fitch@projectcomputing.com

ABSTRACT
We describe a system for automatic post OCR text correction of digital collections of historical texts. Documents, such as old newspapers, are often degraded, so even the best OCR tools can yield garbled text. When keywords are corrupted, text is invisible to search tools. Manual correction is not feasible for large collections. Our non-interactive OCR correction method uses a "noisy channel" approach. The error model uses statistically weighted multiple character edits and a novel visual correlation adjustment using low resolution "reverse OCR". The language model uses normal and also "gap" word 3-grams, plus some 5-grams. Word correction candidates are generated by a deep heuristic search of weighted edit combinations guided by a trie. Testing shows good improvements in word error rate. Experiments demonstrate resilience and justify the use of a deep candidate search.

Categories and Subject Descriptors
H.3.3 [**Information Search and Retrieval**]: Information Filtering; H.3.6 [**Information Search and Retrieval**]: Library Automation - Large text archives; I.2.7 [**Computing Methodologies**]: Artificial Intelligence - Natural language processing; I.5.4 [**Pattern Recognition**]: Applications - Text processing.

General Terms
Algorithms, Design, Experimentation, Performance.

Keywords
OCR, automatic correction, noisy text, historical documents.

1. INTRODUCTION
For some decades there has been massive, expensive, ongoing institutional digitisation of textual resources such as books, magazines, newspaper articles, documents, pamphlets and ephemera from cultural archives. In addition, declassified government documents are being released into the public domain, and many organisations and individuals are converting existing document images into machine readable text via OCR.

The layer of OCR text is the only realistic way for this vast document pool to be exposed to researchers of the world, enabling indexing for search, and access by other natural language processing tools.

DATeCH 2014, May 19 - 20 2014, Madrid, Spain

ACM 978-1-4503-2588-2/14/05 $15.00.
http://dx.doi.org/10.1145/2595188.2595200

Because of problems of physical deterioration, the limitations of scanning and text recognition technology, and the difficulty and expense of quality control in digitisation projects, there is a significant number of documents with a word error rate exceeding 20%, resulting in a "dark pool" of digitised documents poorly represented in search engines.

Simple correction methods, such as comparing against lists of typical OCR errors do not work - even with good OCR, there are too many possible combinations to list. Naive use of a standard spelling checker will not work either. We tried it out of curiosity and got negative results (it made the word error rate worse on our test samples). A specialised approach is needed which can exploit linguistic context to use the redundancy of language (75% character coding redundancy in English [20]), and the predictability of OCR character error patterns to achieve reliable improvements to OCR quality.

We describe an automatic method for processing just the noisy OCR text, and partially repairing that text, using sophisticated models of the document language and the OCR error process (with deep candidate search and a novel word confusion calculator), to increase the visibility of the documents to searching, mining and analysis. The models use statistical language data which can be obtained from a noisy text corpus if needed.

Our goal was to process poor quality OCR text (such as old newspaper) in XML ALTO and hOCR formats using commodity hardware at rates exceeding 1000 words per second, and to at least halve the word error rate.

In experiments, we statistically examine 6512 error pairs from newspaper ground-truth corrections, discovering that correctable words are more likely to be at an edit distance of 4, 5 or 6 than we had expected. We also report on the efficacy of OCR engine character confidence levels which we had attempted to use.

We test our system in several ways, scoring our corrections against ground-truth documents (including documents from a different locale and culture) and comparing error rates with the uncorrected OCR. One series of tests included log-entropy term weighting to estimate effectiveness of correction in a real-world search engine environment. Good results were achieved. We have seen consistent reductions in the word error rate by over 60%, a reduction in search-misses (false negatives) by over 55% and a log-entropy weighted reduction in search-misses by over 50%.

2. ARCHITECTURE
The system is divided into two components - a high level document manager, and a low level correction kernel. The document manager handles document storage and retrieval, threading, XML unwrapping and rewrapping, and maintenance of document level context statistics. After extracting the raw text, it breaks it into blocks to pass to the correction kernel. The document manager pre-processes column breaks and ensures that blocks it passes to the kernel have sensible boundaries. The

document manager draws on word 5-gram data to quickly triage the input, making uncontentious and simple corrections, and identifying text which is very probably correct. It passes to the kernel blocks containing suspect text and its immediate word context, along with information based on a wider context beyond the block: a bag of "topic" words, a parameter representing the noisiness of the text, and some specific word correction suggestions.

The lower level correction kernel uses thread-safe data structures to correct isolated blocks of 15 to 20 blank delimited tokens. It spends most of its CPU time generating suggestions for each of the words in the block it has been passed (subject to triage). It then sorts and culls the suggestion lists, and searches for the best combination (as detailed in the Algorithms section, below) which it returns to the document manager.

3. TRAINING

By "training", we mean acquiring statistics from raw data to build compressed models. This needs multiple data passes, but does not require lengthy cycles of unsupervised learning, nor does it require explicitly labelled data (although the error model can benefit from such data, if available).

The main requirement is a representative text corpus. The tested system is built using an Australian historic newspaper corpus, although we have previously used Wikipedia data.

The first step is to extract 1-gram, 2-gram and 3-gram token frequency lists. Letter case is retained, and tokenization is done by simply splitting on white space. Certain surrounding punctuation is removed from the tokens before use in n-grams.

A fairly crude named entity list (given name, surname or location) is extracted from the raw 3-grams, and can be augmented from any lists available.

The n-grams are stored using numeric codes instead of strings. These codes are defined by our lexicon which is essentially the 1-gram list with a cutoff frequency, filtered by a simple black list of common OCR non-word artifacts. Frequencies are smoothed by fitting to a power law.

The confusion matrix may be re-used on similar corpuses, as it is letter based, but if needed can be adjusted by using available lists of sentence corrections. For a corpus in a different language with roman letters, this weighted edit cost matrix can be approximately adjusted to account for different letter frequencies, then used in "bootstrap" re-training by correcting the corpus, selecting high confidence word corrections, and using them as if they were ground-truth training pairs.

We can also use the various correction tools to process the lexicon to generate a short list of dubious words which can be edited manually and used in a second iteration of training to build a smaller, better lexicon. This list is 1.7% of our lexicon size but was NOT applied, as we were interested in the resilience of the correction process to lexicon errors. So our test results are from a slightly "dirty" lexicon.

The document manager uses 5-grams derived from Google Web 1T 5-gram dataset [4]. Their use is independent of our kernel language model. The 5-grams are available for several languages.

4. ALGORITHMS

The process follows the standard analogy to a "noisy channel" decoding process [19]. From a determination of the probability that a word will appear in the source document (the language model), and a determination of the probability of that word being corrupted into the observed OCR word (the error model) we can apply Bayes' rule and search for a candidate which maximises the probability product. Such OCR correction ideas (hampered by hardware limitations) date back to 1964 [22].

In our approach we use a deep search for candidates which are neighbours of the OCR words, and evaluate them using the error model with the language model.

We apply the language model in 2 stages. The first stage uses a 1-gram model to reduce the size of the candidate lists, and then a 3-gram model is applied combinatorially to pick the best suggestion path with respect to the language context.

Both the document manager and kernel use language models to triage the incoming text, identifying between 60% and 70% of text as "beyond suspicion" and hence not requiring further processing.

4.1 Error Model

An OCR error model is vital in suggesting and evaluating candidates. At the heart of the error model is a confusion matrix giving conditional probabilities of character edits, using both single character edits and edits involving character pairs (such as rn and th). The probabilities can be moderated by a hint as to the prevailing OCR fidelity as observed at the document management level.

The character edit probabilities stored in the confusion matrix are used in several ways. A word can be analysed to estimate its "fragility", the probability that OCR may change it. The matrix can be used to look up the possible correction edits applicable at each letter position of an OCR word, in order of conditional probability, as required by some of our suggestion generators. Finally the matrix can be used to help derive the probability of a false word O being generated by OCR given the assumption that candidate word W is the correct word. We use negative log probability units for this measure, calculated by deriving the most likely weighted multiple character edit path from W to O using a Levenshtein algorithm [6,23] modified to handle statistically weighted character split, pair and join operations as well as the usual substitute, insert and delete. Essentially, the cost of the path is the sum of the character edit costs, including the identity edit which also has a statistically derived cost. Our calculation also incorporates the fact that edits are not strictly independent, as the OCR process tends to be a length preserving text transformation (so excesses of deletes or inserts are penalised).

We call the log measure of O conditional on W the "confusion cost". It is an informal measure of distance from W to O, but it does not define a strict metric space, nor is it a true measure of probability because it derives from the computed probability of the most likely path from W to O, rather than the probability sum over all paths. Nevertheless it is conceptually useful, and works excellently as the error model part of the noisy channel costing.

We can also measure the similarity of W and O with a novel "reverse ocr" procedure. We generate very low resolution glyph bitmaps for the word pair, using a generic font (based on linotype) and overlay the bitmaps in memory, iteratively adjusting alignments to find a reasonable position of correspondence, then calculate a bit correlation measure. This similarity measure has been previously mapped as a confusion cost estimator, using simple regression on a sample of pairs. So in any particular case, in practice, we can get a slightly improved confusion cost by using a linear combination of the matrix derived cost and (with a lower weight), the reverse OCR derived estimate.

4.2 Language Model

The language model is needed to contextually evaluate correction suggestions. We use word 1-grams, 2-grams and 3-grams with Kneser Ney style interpolated backoff [10]. The proportion of unseen n-grams is estimated by fitting a power law distribution. Final backoff from word 1-gram uses character 4-grams plus any available lists of good words held in a Bloom Filter [2], and our named entity list. Numerics are specially treated, and a case variant, and word affix analysis is done to obtain a good probability estimate for non-lexicon words.

The 1-gram lookup is different to the 2-gram and 3-gram, and is incorporated into the lexicon, structured as a minimal directed acyclic word graph [5] which can also yield the extent of partial word match, the number of words with any given prefix, and an indication of whether a full word can be reached or not after the addition of various numbers of characters.

As well as usual left-to-right n-grams, we use a "gap 3-gram" to give the probability of a word conditional on its left and right neighbours, and to identify likely candidates, given the neighbour words. This is used to help generate correction candidates, as a triage tool, and in final checking of the best correction candidate paths.

We also use word 5-grams in the document manager. These are used without smoothing or backoff to help triage "known good" and to generate suggestions (complementary to the suggestions generated in the kernel modules).

4.3 Candidate Generation

We need to generate suggestions before any correction can take place. Suggested words can come from the language model or the error model. Our system allows for multiple suggestors to be "plugged in". An example of when the language model is most useful is in a garbled phrase such as "Antarctic xxxxx sheet". Obviously the suggestion is "ice" and can easily be obtained using word trigrams in a way that does not need further explanation. A different example is "The KvaiiKcllcal press" where the correct suggestion is "Evangelical". This can actually be generated as a top suggestion by our current candidate generator using error model probabilities of the individual edits 'K'->'E', 'ii'->'n', 'K'->'g', 'c'->'e', and 'l'->'i'. It is non trivial and is described below.

The main candidate generator starts with a "seed" word, which is a dubious OCR-generated word, and explores multiple character edit paths which have a promise of leading to a lexicon word within a reasonable probabilistic confusion cost. This search uses the lexicon restructured as a trie to make it easy to track character paths to words. An A* heuristic search [17] is used to cope with the complexity and ensure good paths emerge first.

To prepare for generation from a particular seed, the edits possible at each character position are looked up in the confusion matrix to create a list of coded edits. Each edit code designates: up to 2 emitted characters; the number of seed characters consumed (1, 2, or 3); and the cost (with high cost cut off). The edit cost is the scaled negative log probability of OCR emitting those characters conditional upon consuming the implied seed characters. The identity character edit has a low, but non zero, cost reflecting the character "fragility". There are usually 90 to 100 edits listed per position. This seed structure could also be considered as a weighted finite state automaton with a state for each seed character and the transition arcs labelled by the emitted character(s) and with the weight being the edit cost.

The A* search process traverses the lexicon trie and the seed edit structure. Each step involves popping the partial path with the lowest estimated completion cost off a priority queue, and extending that path by any seed edits possible which are also valid paths from the associated trie position, then for each such extension, calculating the new estimated completion cost and pushing it onto the queue.

The completion cost heuristic we use is mainly based on number of residual characters in the seed, and the 4-gram character cost of that residual path. It is not technically "an admissible heuristic" so there is no guarantee that the first complete path is the best, but as we are gathering many suggestions, that is not a problem - when completed paths are popped off the queue, they are pushed onto a separate output priority queue after modifying the final accumulated path cost by weighting it with the language cost of the lexicon word. Our A* terminates when there is sufficient candidates of quality in the output queue.

The A* suggestion procedure uses the most CPU of the components in our correction system, so it needs to be efficient. We implement it with bit-mapped integer arrays, and use heuristic adjustment, high cost path pruning and duplicate detection. The priority queues are based on min-max heaps [1] so that queued cost ranges can be examined and the heap size kept small for the output queue. The Trie is split into several Tries, each covering different (but overlapping) subsets of the lexicon based on word length. In future we may seek more efficiencies, for example by using a "divide and conquer" approach similar to that in [9].

Because this candidate generation process is guided not by simple discrete edit distance summation, but by probability, it will choose likely 3, 4 or 5 letter edit combinations over unlikely 2 letter edits. However, it should be noted that we cannot guarantee, for example, to output all candidates in a Levenshtein distance of 2. This is because of the nature of our search and also because, as we are using fine grained costing, we effectively perform candidate filtering in the search rather than after having extracted all the neighbours.

Candidates are also generated in other ways, apart from the word mutation method describe above: by splitting and joining at word level, from a "guess" using the left and right neighbouring words and the "gap 3-gram" of the language model, and from suggestions generated and supplied by the document manager. The suggestor collects candidates from the sub-models and adds them to a priority queue according to the sum of their unigram language cost (negative log probability) and their conditional confusion cost (negative log probability of the OCR word given the candidate word). Excessively costly suggestions are culled.

When a word has many low cost suggested candidates, we take it as evidence that the OCR word is in a "crowded environment" with little redundancy and less chance of accurate correction, so we increase the suggestion cost values accordingly.

4.4 Correction

This final correction step is needed because we are automatically correcting. Therefore we need to apply similar judgements that a human operator would make in selecting the best combination of the final culled correction candidates from what can be trillions of possible combinations, using knowledge of language context balanced against the likelihood of the corresponding OCR words deriving from the chosen candidates.

With poor quality OCR, there will be more dubious words, and more candidate lists in a block of words. In addition, the paths

through these lists are not simple, because we must cater for word joins and word splits.

We use a two stage approach for this final language costing. The first stage is to advance a single best word path from left to right, minimising the error and trigram language cost sum at each step but using only the top 3 suggestions for any word.

The second stage is to iteratively apply stochastic variational changes to this initial path, using a simple style of simulated annealing named “record to record travel” and described by Dueck [7]. At each iteration we select and change a candidate and calculate the effect on the total path cost, temporarily accepting if near the best cost, and tracking the actual best path and its cost. The process terminates after a number of iterations based on the number of candidates.

Although the best path will usually include many top ranked candidates, there are many cases where lower ranked suggestions are selected due to lower contextual cost.

5. OBSERVATIONS AND EXPERIMENTS

5.1 OCR confidence ratings

In early development we analysed the ALTO XML CC attribute (character confidence) digits passed on from the OCR engine, trying to fit them mathematically to modify our error model. Unfortunately we gained little benefit, and abandoned their use.

We noted that the small punctuation (dot, comma, quotes) were almost NEVER given low confidence, yet we know they are very unreliable. The letters O,C,E,G,I,c,i,l,1,t,S were usually rated as good, but M,W,g,b,m,w,d rated as bad. We assume letters which attract strong signals on a few feature detectors are given good confidences. Paradoxically, the fact that ‘i’ is highly similar to (and so frequently confused with) a very small cohort of letters, implies higher “confidence” that it will rarely be confused with a much larger set of other letters.

5.2 Resilience to lexicon errors

An effective lexicon needs to be derived from a large corpus similar to the texts to be corrected. Except for very modern texts, this means that the lexicon is likely to contain an unintended set of OCR artifacts, such as “hospltal” or “hononary” which are derived from common words via very common character edits, so that they have a corpus frequency greater than the cut off used in building the lexicon. Such words can be detected in the lexicon by letting our system process its own lexicon, noting words with a correction candidate having a very small confusion cost and a significant language cost difference. The reason this can work is that we have a fine grained confusion cost.

We automatically flagged 1.7% of our test lexicon word types as dubious (possibly OCR artifacts). However, due to the fact that common words, and especially short words, have many close orthographic neighbours, the OCR artifacts can only be practically removed from the lexicon by including a human in the loop. This is simple, but we decided to leave these OCR artifacts in the lexicon for our tests. It had no real effect on performance, because the suggestor easily generated the correct word and the noisy channel logic selected that correct word.

Of course, the nature of language itself implies that no lexicon can be perfect, but we can take comfort that most imperfections lie in the extreme low frequency long tail, and do not have much impact on correction performance when using frequency based n-grams.

We note the need to derive the correction lexicon from a homogeneous relevant corpus, as a corpus containing both modern and historical word spelling variants may yield infelicitous corrections to the more frequent modern spelling form, if the spelling edit also happens to be a very common OCR edit.

5.3 Empirical study of errors

We have a mechanism where words in context can be judged to select the best combination of correction candidates by balancing the error and language model costs. But good correction is impossible without a good candidate generator.

Our main candidate generator takes a possibly corrupt OCR word and searches for a "close" lexicon word. Because this can be very CPU intensive, we wanted to study some OCR errors to find out the "close enough" point to bound the candidate search.

So we extracted every word error pair from OCR aligned with ground-truth newspaper text. We considered all word to word correction pairs, where neither word contained blanks, and removed normal word punctuation (brackets, quotes, commas, periods etc). We retained all such word pairs, including numerics, stranded punctuation, uncorrectable garbage, corrections to non-lexicon words, and even a few bad pairs resulting from alignment error or imperfect "ground-truth". This yielded 8991 token pairs (6512 OCR word types). We used our candidate generator in stand-alone mode to generate up to 12 candidates per OCR token.

To illustrate why we do not attempt correction by using standard error lists, we note that the simple word 'the' had 95 error types even in our relatively small sample, with 28 of them being seen once only.

The spelling correction literature, as surveyed by Kukich [14], has various estimates of the closeness of correction pairs. Damerau [6] suggested that 80% of spelling errors are one edit [6, 23] away from the correct word. However, spelling and OCR errors differ. OCR engines are not tricked by rhyme, rhythm, reason, grammar or keyboards, but are affected by visual noise bursts. Tanner [21] showed a high character error rate of 16.4% and word error rate of 22% in 19th century newspapers, which (given the average word token length of 5) is suggestive of error bursts - multiple character edits per word.

We will use E1, E2, E3 etc to denote an edit distance (number of character substitutions, deletions and insertions). We found in our sample that 90% of the pairs were separated by E3 or less. If the residual 10% were "hopeless cases" we could bound our search at E3 with a saving of CPU time. However, we found that for 46.8% of these residual "bad" words, the correct suggestion was generated. And in a surprising 30.8%, the top ranked suggestion was correct at E4, E5 and E6. Even at E6, we observed 21 correct top ranked suggestions for the 99 OCR tokens at E6.

The easy pickings at E4, E5 and E6 were usually longer words which are less affected by multiple edits as they have fewer orthographic neighbours that they can be confused with. They are also more valuable words, likely to make good search terms, and so are worth correcting.

Simply using E6 as a fixed bound on candidate search is not viable as most tokens are less than 6 letters long (with a median between 4 and 5 letters). Candidate lists based on such an approach would be enormous.

This is why we chose an A* search of the lexicon for our candidate generator, using a fined grained probabilistic measure of editing cost, with character edit costs derived in advance statistically. The candidates emerge in approximate probability order, and likely E4, E5 and E6 candidates will therefore appear

before unlikely E1 and E2 candidates, and we stop the search when enough "good" candidates have emerged (between 1 and 20).

To illustrate the quality of suggestions, Table 1 shows OCR and the top suggestion (which is an "easy" correction) at E6, E7 and E8, all of which are very difficult to guess.

Table 1. Example E6, E7 and E8 suggestions

OCR	Top Suggestion
Parhumuitar}	Parliamentary
I.iulwuvB	Railways
Itegtniont	Regiment
niltfltory	adultery
uj.rccu.eut	agreement
couniutfc.o	committee
cnuipuii	company
dctoimiuatJOu	determination
uiidcrtikcr'a	undertaker's

6. TESTS AND RESULTS

6.1 Datasets

Raw OCR text from a relevant corpus, paired with ground-truth text, is needed. It is hard to find good ground-truth, and it is tempting to start with good text and artificially synthesise a degraded image for OCR, but we wanted real data, preferably from early newspapers processed by ABBYY FineReader, as that constitutes the bulk of material.

We attempted an evaluation on 3 datasets:

1. "Mostly-corrected" medium length articles from the Sydney Morning Herald, 1842-1954. These yielded a large amount of text, but as we discovered, poor quality ground-truth.

The National Library of Australia's Trove Digitised newspapers [16] site contains over 10 million OCR'ed newspaper pages. Each page is zoned into its component articles. The public can correct the OCR [12], and as of March 2014, over 120M column-lines of text have been corrected. Although this seems a vast amount, only a small percentage of articles have any corrections [11].

From the NLA newspapers web site, it is possible to retrieve for an article both the most recent version of the text (incorporating all corrections made to-date) and a complete history of corrected line pairs (showing the text before and after the correction and the date of the correction). Hence, by obtaining a copy of the current version of the text and then by backing-out corrections in reverse-date order on that copy, it is possible to recreate the original text, as OCR'ed. For this dataset we chose medium length (between 100 and 1000 word) news articles having a total number of line corrections of at least 85% of the number of lines in the article. Note that this does not mean that the entire article or even 85% of it has been corrected, as a single line corrected 85 times in a 100 line article would meet this 85% criteria, but it does serve to identify articles which humans have intensively corrected.

It soon became apparent that is it not possible to treat even extensively corrected versions of articles as ground-truth. Amongst the problems: (i) Not all lines with errors have been corrected. Even with the 85% criteria, correctors often leave a heading or final paragraph uncorrected, or just miss 'obvious' errors in a line and don't correct the line. Human corrected lines often contain uncorrected errors such as "e" erroneously OCR'ed as "c" (as in "Thc") and "l" OCR'ed as "I" (as in "ltaly"). (ii) Line boundaries are changed. Some correctors occasionally move text across line boundaries for no apparent reason. Sometimes this is minor, such as combining both parts of a hyphenated word on its starting line and removing the hyphen. Other times, it is less predictable, and extensive. (iii) Words are changed. Quite commonly names formed as, for example "M'Donald" are corrected as "McDonald". Occasionally, newspaper typos are corrected, and words are deliberately changed rather than corrected. (iv) Content is added. Frequently the OCR and zoning process misses text at the start or end of line, entire lines, or a paragraph, typically at the end of an article.

Such issues create many problems when attempting to evaluate the performance of an OCR correction process, both because the human ground-truth is not accurate and because the raw OCR is totally missing some content that appears in the ground-truth. Due to the poor quality of the ground-truth in this dataset, we abandoned our attempt to use it for performance evaluation.

2. Completely corrected medium-length articles from the Sydney Morning Herald, 1842-1954. A randomly selected subset of Dataset 1 containing 49 thousand words in 159 articles additionally corrected by us to ground-truth.

3. Article text from the Library of Congress Chronicling America newspaper archive. A randomly selected sample of 18 thousand words from 49 articles from 5 U.S. newspapers manually corrected by us to ground-truth.

All 3 datasets are available for download, linked from the overProof evaluation dataset web site [18].

6.2 Methodology

We corrected the text as OCR'ed using the methods described in this paper, giving us three versions of the text for each article: (i) text as OCR'ed, (ii) ground-truth version of this text, produced by human correction of the OCR text, and (iii) automatically corrected text.

We performed three measurements on each dataset:

1. Recall improvement. Finding an article in a search engine relies on the indexing of the correct article text. This measurement calculates the reduction in search misses achieved after article text correction. It does this by comparing the number of unique ground-truth words found amongst the OCR text with the number found amongst the corrected text.

2. Readability improvement. The readability of article text is determined by the accuracy of the digitised text. This measurement calculates the reduction in the number of erroneous words after article correction. Note that whereas the recall improvement measurement requires that only one of possibly several occurrences of a ground-truth word be accurate, this measurement considers how many occurrences are correct.

3. Weighted-recall improvement. Similar to the first measurement, but attempts to also weigh words based on their entropy, placing a higher value on the correction of words whose appearance in a search index is more likely to result in more relevant documents being returned to a searcher [8].

Testing was performed as follows:

1. Words are extracted from each version by splitting the text on white space. Leading and trailing punctuation are discarded. Single character words, words starting with currency symbol or

containing numbers and other non-alphabetic characters (other than hyphen and apostrophe) are discarded. Hyphenated words are changed to their non-hyphenated form. Words are changed to their lower-case form.

2. The entropy of ground-truth words is calculated for the dataset.

3. A set of unique words appearing in each version (original OCR, ground-truth and corrected) is created, recording the number of times they appear in each version.

4. Words from the ground-truth version are checked for appearance in the original OCR and corrected version. For the recall measurement, only presence or absence is noted. For the readability/word-correction measurement, the number of occurrences of words is counted. For the weighted-recall measurement, the log of the occurrence count of the word in the article is multiplied by the word document entropy.

6.3 Results

The results of tests on Datasets 2 and 3 are shown in Table 2 and Table 3 respectively.

We plotted reduction in recall misses against uncorrected recall for each article in Dataset 2 (Figure 1) and recall for each article in Dataset 2 before and after correction (Figure 2). Equivalent graphs for Dataset 3 show similar distributions of good reductions in recall misses across a range of OCR'ed text qualities.

Table 2. Recall and word error for Dataset 2

Measurement	Uncorrected	Corrected	Improvement
Recall misses	16.2%	6.6%	59.3%
Word Error rate	18.5%	6.3%	66.0%
Weighted recall misses	16.2%	7.1%	56.0%

Table 3. Recall and word error for Dataset 3

Measurement	Uncorrected	Corrected	Improvement
Recall misses	16.0%	6.7%	57.7%
Word Error rate	19.1%	6.4%	66.5%
Weighted recall misses	16.0%	7.4%	54.0%

7. RELATED WORK

Because we are describing a complete end to end correction system, there is related work in the fields of spelling correction, speech recognition, statistical language modelling, DNA matching, and search algorithms which is too numerous to list. In the specific field of approximate search for candidates in lexica, there are some other relevant approaches. Gerdjikov, Mihov, Mitankin and Schulz [9] describe a highly optimised method of extracting all candidates at a given Levenshtein edit distance. Huldén [13] describes another approach using A* search. Boytsov [3] surveys the state of the art in this area as at 2011.

8. COMPARISON WITH HUMAN PERFORMANCE

Anecdotally, a human corrector (HC) is able to achieve high quality results, although individual HC units differ greatly. An HC takes years of expensive training, but even then has a mediocre

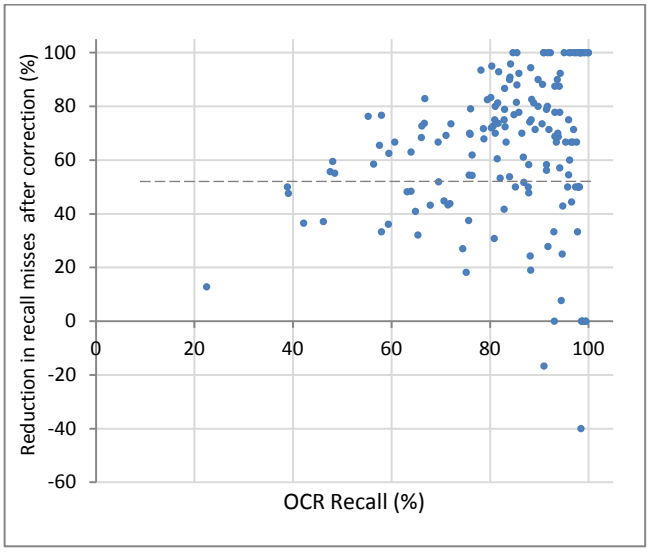

Figure 1. Reduction in recall misses

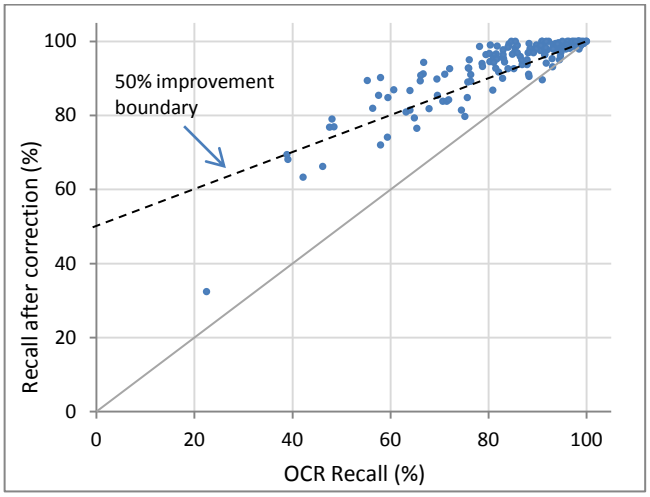

Figure 2. Recall before and after correction

vocabulary, much less than the size of our lexicon. A typical HC is poor on character by character checking and can miss very obvious errors, especially involving the letters i, l, I, 1, 0, O, o, e, c.

Although expensive, there is a good supply of HC units. Unfortunately, quality is variable and performance is context dependent - topic areas such as football or science or knitting show large individual differences in HC performance. Ideally, HC units should be tested before use. As HC units are analog in nature rather than digital, they cannot deal directly with the digitally stored text data but need devices to bridge the "analog gap" which lowers the HC correction rate. Maximum observed sustained rate for an HC on Australian newspapers [16] is about 8 words per minute, and appeared to be limited mainly to the specialised area of genealogy. By comparison, software such as ours can process 10,000 times faster on a single commodity server, and this rate can be arbitrarily increased by adding extra hardware. Note also that while silicon based computer hardware performance increases annually, the basic HC hardware performance has not changed in millennia. Multiple HC units can be used in parallel to correct a

corpus, but the work rate can drop as correction coverage increases. This is due to "selective topic attachment" and the fact that individual HC units can disengage when the availability of uncorrected text articles in that topic dwindle to low values.

Humans may be slow, but they have a deep understanding of language, which is especially required in the final step in automated text correction - judging the best sentence composed out of the automatically generated word correction candidates. This ability can be measured by the Microsoft Research sentence completion challenge [24], the state of the art solution being a recurrent neural network [15]. This could point to the future of automatic OCR correction but unfortunately requires (as with humans) a long training time and a large amount of training data (which may be unavailable in historical text).

9. CONCLUSIONS

Searching or mining a digitised text corpus can be fundamentally limited by the inconsistent quality of OCR text.

We described an unsupervised OCR correction system using statistically weighted multiple character edits, novel visual correlation adjustment, a high context language model, and multiple candidate generators, including a deep best-first candidate search.

Our experiments confirmed that quality candidates can be generated at large edit distances.

The implementation was tested against ground-truth historic newspaper text from several sources, and delivered a reduction in word error rate of over 60% and a reduction in search-misses of over 50%. While gathering ground-truth, we noted that even articles corrected by humans are far from perfect.

Our system is a fully functional end to end batch OCR corrector delivering corrected texts at a high rate on a standard commercial "cloud" server. We are satisfied with the principles of the approach and we are currently investigating minor modifications to further enhance correction performance.

10. REFERENCES

[1] Atkinson, M. D., Sack, J. R., Santoro, N., & Strothotte, T. (1986). Min-max heaps and generalized priority queues. *Communications of the ACM*, 29(10), 996-1000.

[2] Bloom, B. H. (1970). Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7), 422-426.

[3] Boytsov, L. (2011). Indexing methods for approximate dictionary searching: Comparative analysis. *Journal of Experimental Algorithmics (JEA), 16*, 1-1.

[4] Brants, T., & Franz, A. (2009). Web 1T 5-gram, 10 European languages version 1. *Linguistic Data Consortium, Philadelphia*.

[5] Daciuk, J., Mihov, S., Watson, B. W., & Watson, R. E. (2000). Incremental construction of minimal acyclic finite-state automata. *Computational linguistics*, 26(1), 3-16.

[6] Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3), 171-176.

[7] Dueck, G. (1993). New optimization heuristics: the great deluge algorithm and the record-to-record travel. *Journal of Computational Physics,* 104(1), 86-92.

[8] Dumais, S. T. (1991). Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments, & Computers*, 23(2), 229-236.

[9] Gerdjikov, S., Mihov, S., Mitankin, P., & Schulz, K. U. (2013). Good parts first-a new algorithm for approximate search in lexica and string databases. *arXiv preprint arXiv:1301.0722*.

[10] Goodman, J. T. (2001). A bit of progress in language modeling. *Computer Speech & Language*, 15(4), 403-434.

[11] Hagon, P. (2013) Trove Crowdsourcing Behaviour. In *Australian Library & Information Association Information Online 2013 Proceedings*. Retrieved from http://www.information-online.com.au/pdf/Tuesday_Concurrent_2_1125_Hagon.pdf

[12] Holley, R (2009). How Good Can It Get? Analysing and Improving OCR Accuracy in Large Scale Historic Newspaper Digitisation Programs. *D-Lib Magazine*, 15(3/4), 1082-9873.

[13] Huldén, M. (2009). Fast approximate string matching with finite automata. *Procesamiento del lenguaje natural*, 43, 57-64.

[14] Kukich, K. (1992). Techniques for automatically correcting words in text. *ACM Computing Surveys* (CSUR), 24(4), 377-439.

[15] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781.*

[16] National Library of Australia (2014, January 5). *Trove Digitised Newspapers.* Retrieved from http://trove.nla.gov.au/newspaper

[17] Och, F. J., Ueffing, N., & Ney, H. (2001, July). An efficient A* search algorithm for statistical machine translation. In *Proceedings of the workshop on Data-driven methods in machine translation* -Volume 14 (pp. 1-8). Association for Computational Linguistics.

[18] Project Computing (2014, January 5). *OverProof Evaluation Data*. Retrieved from http://overproof.projectcomputing.com/datasets/

[19] Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27(7,10), 370-423, 623-656.

[20] Shannon, C. E. (1951). Prediction and entropy of printed English. *Bell system technical journal*, 30(1), 50-64.

[21] Tanner, S., Muñoz, T., & Ros, P. H. (2009). Measuring mass text digitization quality and usefulness. *D-Lib Magazine*, 15(7/8), 1082-9873.

[22] Vossler, C. M., & Branston, N. M. (1964, January). The use of context for correcting garbled English text. In *Proceedings of the 1964 19th ACM national conference* (pp. 42-401). ACM

[23] Wagner, R. A., & Fischer, M. J. (1974). The string-to-string correction problem. *Journal of the ACM* (JACM), 21(1), 168-173.

[24] Zweig, G., & Burges, C. J. (2011). The Microsoft Research sentence completion challenge. Technical Report MSR-TR-2011-129, Microsoft.