Approximate String Matching Algorithms for Limited-Vocabulary OCR Output Correction

Thomas A. Lasko[a], Susan E. Hauser[b]

[a]Gundersen Lutheran Medical Center, La Crosse, WI 54601

[b]Lister Hill National Center for Biomedical Communications, National Library of Medicine, Bethesda, MD 20894

ABSTRACT

Five methods for matching words mistranslated by optical character recognition to their most likely match in a reference dictionary were tested on data from the archives of the National Library of Medicine. The methods, including an adaptation of the cross correlation algorithm, the generic edit distance algorithm, the edit distance algorithm with a probabilistic substitution matrix, Bayesian analysis, and Bayesian analysis on an actively thinned reference dictionary were implemented and their accuracy rates compared. Of the five, the Bayesian algorithm produced the most correct matches (87%), and had the advantage of producing scores that have a useful and practical interpretation.

Keywords: approximate string matching, optical character recognition, spell checking, Bayes' theorem

1. INTRODUCTION

The National Library of Medicine (NLM) maintains the medical article citation database MEDLINE®[1]. Semi-automatic citation entry into MEDLINE is facilitated by a system developed by NLM's Communications Engineering Branch that uses optical character recognition (OCR) of the scanned first page of each article. Data from the OCR conversion is used along with image analysis of the scanned page to identify those areas of the page corresponding to the title, authors' names, authors' affiliations and the abstract, and text from those areas is assigned to corresponding database fields of the MEDLINE citation entry. Following this automated process, each preliminary citation is corrected and verified by reconcile operators. The affiliations field of the citation is uniquely difficult to process automatically because it is often printed in italics or a small font (or both), and the quality of the character recognition for that field is frequently poor, with many incorrect, added, or deleted characters.

The initial OCR algorithm associates a confidence level with each of its output characters, with levels ranging from 1 (low confidence) to 9 (high confidence). The confidence levels have been found to be a roughly monotonic representation of character translation accuracy rates, with confidence 9 being correct more than 99% of the time, and confidence 1 correct about 15% of the time. Words with characters all of confidence 9 are assumed correct and are not processed further. Words containing at least one character of confidence 1 through 8 are verified, corrected, or retyped. The reconcile operators report that up to 20% of the affiliations fields contain roughly 30% low-confidence characters, and they have found that the most efficient correction of such fields requires retyping them in their entirety. In many instances words include low confidence characters but are correct, and the operator must spend valuable time verifying and marking them as correct.

The affiliations field is further unique in that many of the words that appear in the field come from a limited vocabulary. These include institutional titles like "Department," "University" and "Institute," biomedical profession references like "Pathology," "Medicine" and "Molecular" plus the names of cities, states and countries.

The OCR engines use a proprietary, third party algorithm that is not available to NLM for modification, so any improvement in the translation must be done as post-processing. In this paper we describe five different word-matching schemes that are well suited to correcting these types of errors, taking advantage of the limited vocabulary. The schemes were implemented and their accuracy rates experimentally determined.

2. METHODS

OCR output strings and their human-corrected, final MEDLINE citations were obtained from NLM archives. Two ground truth tables were constructed from this data (one with 22,000 entries constructed by computer and one with 15,000 constructed by hand), pairing each low-confidence word in the raw OCR output with its corresponding human-corrected word, and deleting any OCR words that were not ultimately included in the citation entry. The entries in this table were then compared character by character to construct a substitution matrix containing the frequencies of translation, addition, and deletion for each scanned character presented to the OCR algorithm.

A reference dictionary of 86,000 correct words and their frequencies of occurrence in the affiliations field was constructed from a separate set of archived data, and five different methods were used to match the raw OCR words in the ground truth table with their best dictionary match. Those dictionary matches were then checked against the true word match in the ground truth table, and the matching methods were scored by the percentage of matches that were identical to the known true word.

2.1. Definitions

In this paper, the raw, unverified OCR output will be referred to as 'OCR' words and letters, and the final, human corrected entries as 'true' words or letters. An OCR word with at least one character of confidence less than 9 will be called a "low-confidence word", and one with characters all of confidence 9 will be called a "high-confidence word".

2.2. Ground truth tables

The human-constructed ground truth table was produced by presenting human users with an entire affiliations record, with its OCR and true data simultaneously on the screen. The low-confidence OCR words were highlighted one by one, and the user selected the corresponding true word with a mouse click in the true data section. This pair of words and their corresponding confidence levels were then stored as one line in the truth table, and the process repeated for every low-confidence word in the OCR fields. This produced a truth table of 15605 entries.

The machine-constructed ground truth table was produced in a similar manner, but with an algorithm selecting the word matches. The edit distance algorithm described below was used at the letter level to discern whether the words were approximate matches, and again at the word level to align words within the paragraphs. If the word pairs were certain matches by position alone, (*i.e.* high-confidence matches on either side) they were added to the truth table regardless of how good of a match they were on a letter basis. If one word in one section had a possible mapping to several words in the other section (due to deletions or insertions in the final entry), then a pair was added to the truth table if it was the best match of the set and the match was greater than 58% (an empirically chosen value to minimize false positives and false negatives). If the match was between 25% and 58%, it was output to a second file which was checked by hand and correct matches added to the ground truth table. If a pair matched to less than 25% (again empiric), the entry was written to a third file which was checked by hand for possible matches missed. This process produced a truth table of 21800 entries.

These tables were then used to test the various matching algorithms. Each algorithm was presented sequentially with the OCR words from the ground truth table, and the algorithm's dictionary match was compared with the true word in the table to determine the accuracy rate.

2.3. True word dictionary

The dictionary was extracted from a set of MEDLINE citation data separate from that used to construct the ground truth tables. Human corrected words were extracted and counted by machine, using spaces and some punctuation marks as word boundaries. Several dictionaries were constructed, with various parametric differences. The one ultimately used consisted of about 86,000 upper-and-lower case words of length > 2 plus the 24 highest-frequency words of length = 2.

2.4. Substitution matrix

A 2-dimensional matrix was constructed using the entries in the machine-derived ground truth table. The letters of an OCR word was aligned with those of its true word using the letter-wise edit distance algorithm (see below). This alignment revealed all correct letter translations by the OCR algorithm and all letter substitutions, deletions and insertions. The number of occurrences was counted for deletions of true word characters, for insertion of new OCR characters, and for translation of every character in the true text character set to every character in the OCR character set. The probability of substitution $P_M[x,y]$ of each OCR letter y for every true letter x (including the case where $x = y$), the probability of deletion $P_D[x]$ of each true letter x, and the probability of insertion $P_I[y]$ of each OCR letter y were calculated as follows:

$$P_M[x,y] = \frac{\sum (x \to y \text{ translations})}{\sum \text{all } y \text{ in OCR strings}} \tag{1}$$

$$P_D[x] = \frac{\sum (x \to < \text{none} > \text{translations})}{\sum \text{all } x \text{ in true word strings}} \tag{2}$$

$$P_I[y] = \frac{\sum (< \text{none} > \to y \text{ translations})}{\sum \text{all } y \text{ in OCR strings}} \tag{3}$$

The matching algorithms then used this matrix of letter translations to calculate the various word-match scores.

2.5. Cross correlation matching method

The first matching method attempted was an adaptation of the cross-correlation function. The cross-correlation function is commonly used in image and signal processing where an unknown signal is searched for a known feature or shape, and is sometimes described as a "sliding dot-product". In this project, the cross-correlation function was modified to operate on words and letters instead of quantitative signals. The function compares words a (of length m) and b (of length $n \geq m$) and produces a vector W of length $l = m + n - 1$ with elements W_i (where $i = 0 \dots l - 1$) defined by equation (4):

$$W_i(a,b) = \begin{cases} \sum_{j=0}^{i} S(a_j, b_{(n-1)-i+j}) & \text{if } i = 0 \dots m-1 \\ \sum_{j=0}^{m-1} S(a_j, b_{(n-1)-i+j}) & \text{if } i = m \dots n-1, m \neq n \\ \sum_{j=0}^{(l-1)-i} S(a_{(n-1)-(l-1-i)+j}, b_j) & \text{if } i = n \dots l-1 \end{cases} \tag{4}$$

where

$$S(x,y) = \begin{cases} 1 \text{ if } x = y \\ 0 \text{ if } x \neq y \end{cases} \tag{5}$$

This function compares the two words by aligning them against each other and examining the corresponding pairs of letters. The element W_0 is defined as the alignment position where the last letter of word b aligns with the first letter of word a.

Example calculations using the true word "`Biology`" and the mistranslated OCR word "`1Biologv`" are shown in Table 1 below:

Vector Element	W_0(`Biology, 1Biologv`)	W_4(`Biology, 1Biologv`)	W_6(`Biology, 1Biologv`)

Word alignment	`       Biology`
	`1Biologv`
	`       0`
		`    Biology`
		`1Biologv`
		`    00100`
			`  Biology`
			`1Biologv`
			`  1111110`

Score	0	0+0+1+0+0=1	1+1+1+1+1+1+0=6

Table 1 - Example calculation of cross-correlation vector elements

The full vector $\boldsymbol{W}$ (`Biology, 1Biologv`), then, is <0,0,0,0,1,0,6,0,1,0,0,0,0,0>.

If there are any matching letters at all between the two words there will be some nonzero elements in *W*, though they will be randomly arranged if the words are unrelated. If the word correlation is perfect and the words contain no repeated letters, there will be a single non-zero element, or 'peak', of height *n* at W_{n-1}. If it is a perfect match except for one substituted letter, the peak will be the same width but one count lower in height. An otherwise good correlation but with one or more inserted letters in the middle of one of the words will result in several smaller peaks that together sum to *n*. A poor correlation will result in several small peaks scattered throughout the vector, or perhaps a zero vector.

An advantage of this function is that it can be extended to take into account OCR mistranslation rates by using the frequency of substitution for any character *x* to any other character *y* as the value of *S* instead of simply 0 or 1. In the example above, the frequency of an OCR letter `v` coming from a true letter `y` with our OCR engine might have been about 0.2. So S(`v`,`y`) = 0.2, instead of 0. This extension would allow a more precise and sensitive correlation between two candidate words.

Because of the way this function was used to rank possible word matches, a single numeric score representative of the quality of match needed to be extracted from the cross-correlation vector *W*. In image and signal processing, the value and position of the peak are the parameters usually used. In this case, much more information can be gleaned by studying the entire vector as described above to detect single inserted elements or other common mistranslations. The processing of the vector could have been done in various ways, and investigation into the optimum method may be a rich source of future improvement for this algorithm. In this project the height of the peak over the width was taken as the measure of the match. The width *D* of the peak was calculated as:

$$D = \sqrt{\frac{\sum_i W_i \left(x_i - \bar{x}\right)^2}{\sum_i W_i}} \qquad (6)$$

where

$$\bar{x} = \frac{\sum_i W_i x_i}{\sum_i W_i} \qquad (7)$$

and the x_i are simply the indices of W_i, (i.e. $x_i = i$).

The vector signature of a perfect match will vary depending on the number and position of repeated letters in the word, and when these different signatures are scored with (6) and (7), those scores will vary considerably, even though the match is perfect in each case. To normalize against this variation, every match of an OCR word with a dictionary word was divided by the correlation of the dictionary word against itself, giving a final score in the range of zero to one, with one indicating a perfect match. This normalization proved effective in accounting for different word structures, and word matches with similar scores usually appeared subjectively equally well-matched.

2.6. Edit distance matching method

The edit distance algorithm is a well-known recursive algorithm[2] which supplies the optimum alignment between two ordered strings and determines the global minimum number of edits needed to transform one string into the other. In this algorithm the edits can be insertions, deletions, or substitutions. The algorithm is often implemented using a dynamic programming approach that calculates the number of edits *D* between every possible left-sided substring of each of the two words *a* and *b*. $D(a_i, b_j)$, for example, is the edit distance between the first *i* letters of the word *a* and the first *j* letters of the word *b*. The dynamic programming calculation is recursive, with

$$D(a_i, b_j) = \min \begin{pmatrix} D(a_i, b_{j-1}) + C_I(b_i) \\ D(a_{i-1}, b_{j-1}) + C_M(a_i, b_i) \\ D(a_{i-1}, b_j) + C_D(a_i) \end{pmatrix} \quad (8)$$

where C_I, C_M, and C_D, are the costs of insertion, match/substitution, and deletion respectively. In the simplest case, the costs of insertion, deletion, and substitution are all unit costs, and the cost of a match is zero. That is, $C_I(x) = C_D(x) = 1$ for all *x*, and $C_M(x,y) = 0$ if $x = y$ and 1 otherwise.

If the two words *a* and *b* have lengths *m* and *n*, then $D(a_m,b_n)$ will contain the minimum edit distance needed to transform *a* into *b*. In our use of this function, normalizing by word length was desirable, so the final matching score was determined by D/*m*, giving the edit distance as a percentage of the true word length.

The method can be extended to use different costs for different edits, so that $C_M(x,y)$ may depend on the frequency of the OCR substitutions of *x* for *y*, $C_I(x)$ on the frequency of inserting character *x*, and $C_D(x)$ on the frequency of deleting character *y*, in a similar way to that described for the cross correlation method above. This is commonly implemented using a substitution matrix as described above. For this study, the edit distance algorithm was implemented first without and then with a substitution matrix. The implementation with a substitution matrix is described in the next section.

2.7. Probability of match method

The probability of match method is the edit distance method with the addition of the probabilistic substitution matrix. The edit distance algorithm was implemented slightly differently with this method in order to directly calculate the probability that a true word *a* would be mistranslated into the OCR word *b*. In order to calculate this result, the probability of the OCR making each edit was looked up in the substitution matrix, and all of the letter-wise probabilities were multiplied together to form the word translation probability. This implementation also required modifying the algorithm to find the maximum probability of translation rather than the minimum number of edits. The recurrence relation became, then

$$D(a_i, b_j) = \max\begin{pmatrix} D(a_i, b_{j-1}) \cdot P_I(b_i) \\ D(a_{i-1}, b_{j-1}) \cdot P_M(a_i, b_i) \\ D(a_{i-1}, b_j) \cdot P_D(a_i) \end{pmatrix} \qquad (9)$$

where $P_I(x)$ and $P_D(x)$ are the frequency of insertion and deletion of the character x, and $P_M(x,y)$ the frequency of substituting x for y.

2.8. Bayesian probability matching method

The Bayesian probability matching method further extends the edit distance algorithm, and considers the probability of match with the dictionary word (using the above probability function), the frequency of occurrence of the dictionary word and the probabilities and frequencies of *all the other* dictionary words when calculating the ranking score. The basic Bayes' equation is:

$$\mathrm{P}(t|o) = \frac{\mathrm{P}(o|t)\mathrm{P}(t)}{\mathrm{P}(o)} \qquad (10)$$

Where:

P(t|o) is the probability that the true word is t given that the OCR word is o.
P(o|t) is the probability that the OCR will output the word o when presented with the dictionary word t. (This is the probability of match function above.)
P(t) is the frequency of the word t in the dictionary, and
P(o) is the frequency of the word o in the OCR output space.

It may at first seem problematic to find P(o), which would need a huge database of OCR output to directly count the frequencies of each unique mistranslation. Fortunately, the quantity can be calculated by summing the probability of producing o from each true word in the dictionary, weighted by the probability of that true word occurring in the affiliations text. In mathematical terms,

$$\mathrm{P}(o) = \sum_i \mathrm{P}(o|t_i)\mathrm{P}(t_i) \qquad (11)$$

where the t_i are the words in the dictionary, and the sum is over all entries. It turns out that this is not even computationally expensive: $\mathrm{P}(o|t_i) \cdot \mathrm{P}(t_i)$ is already being calculated for every [o, t_i] pair since it is the numerator in equation (10), and it is only a matter of keeping a running sum as the function loops through all t_i in the dictionary.

2.9. Bayesian method with dictionary thinning

In an effort to make the calculation more efficient a previously written word matching function[3] was used to reduce the dictionary to several hundred high-probability candidate words before processing with the Bayesian function. The thinning function worked by examining the confidence values for the OCR word and selecting bigrams from the word that have the highest confidence values. It then created a reduced dictionary of only those words that contain the selected bigrams. This dictionary was then ranked using scores produced by the Bayesian method.

3. RESULTS

The ranking method accuracies are given in Table 2. The third column contains the overall accuracy rates, and the fourth and fifth columns contain adjusted accuracy rates. The adjustments were done in order to compare one method against another, with the effect of the completeness of the reference dictionary removed. Since the dictionary and the ground truth test set were generated from separate sets of data, they inevitably each contain some words that do not appear in the other. This was intended, since it more accurately reflects the conditions of eventual use of the algorithms in data correction. The overall accuracy rates reflect the percentage of words the algorithms correctly matched, including those words that do not overlap with the reference dictionary. For comparisons between the various matching methods, however, the accuracy rate was adjusted so that those words not found in the reference dictionary were removed from the scoring, since every method will fail to match them. The adjusted accuracy score gives the measure of how often a word will be matched when the correct word is in the dictionary to begin with, and will be generally a higher rate than that achievable in practice, though with optimization of the reference dictionary, this score can be approached.

Method	Ground Truth Table Type	Overall Accuracy (%)	Accuracy Adjusted for Dictionary (%)	
Edit Distance	**Machine**	**77.3**	**87.6**	
Cross-correlation	**Machine**	**78.5**	**89.0**	
Probability Function	**Machine**	**85.0**	**96.3**	
Bayes	**Machine**	**87.0**	**98.6**	
Bayes with Thinning	**Machine**	**85.7**	**97.1**	**99.1**
	Human	**77.8**	**93.6**	**97.6**

Table 2 - Ranking method accuracy rates

4.DISCUSSION

In the problem of correcting OCR output, the question is, ‘Given this OCR word, what is the probability that it was produced by that particular dictionary word?’ Of the various methods described, the Bayesian method is the only one that truly estimates this probability, and as can be seen, it gave the best results in our tests.

4.1. Edit distance and cross correlation

The edit distance and cross-correlation functions provide a value that is more of a match *index* than a match *probability*. That is, a match with about half the letters in common would earn a score of about 0.5 from both of these methods, yet the probability that one came from the other is much lower than 50% if half the letters are different, considering that the average is something like 10% per letter rate of substitution. This is only problematic if further calculation is desired, such as in the Bayesian calculation.

These two methods produced very respectable accuracy rates, even though they made no use of the OCR characteristic substitution rates, and used generic costs for substitution, deletion, etc.

A strength of the cross-correlation method is it gives inherent position data. All of the methods used the space character to define word boundaries, but by using the peak position information the cross-correlation method could most easily detect an inserted space. In this event the peak of the vector *W* would occur near one end of the vector instead of near the center, and the word neighboring that end could then be examined for a corresponding but mirrored peak indicating two halves of a word matching with the reference. The edit distance method might also be modified for similar analysis by keeping track of inserts and deletes separately, with a run of deletes neighboring a run of inserts indicating a possible word split, but this would be more cumbersome than simply examining the cross correlation vector.

4.2. Probability of match

The probability of match function uses the substitution data but not the word frequency data provided by the dictionary. Using the substitution data did improve performance over the similar but generic edit distance algorithm, with accuracy of 85% overall.

4.3. Bayesian match function

This method takes the probabilistic edit distance method one step further, including word frequency information in its calculations. As expected, it gave the best results, though by a surprisingly low margin. Comparing it with the probability of match function shows an improvement of about 2 percentage points coming with use of the frequency data. Since use of this data is computationally inexpensive, however, there is no reason to exclude it.

The Bayesian match function returns a number indicative of the percentage of time one would be correct in making that word substitution. It takes into account the closeness of all the matches in the dictionary before it computes the final word match scores. In practice, most of the correct matches returned scores between 0.9 and 0.999999. These scores have the useful property that their sum over all words tested equals one. This means, for instance, that if one wanted to keep the accuracy rate for all substitutions above any given level, say 99.9%, then if the score on a certain match is greater than 0.999, the correction may be made without reservation, and if the highest score is less than 0.999, the next highest matches are added to the match possibilities until their sum is greater than 0.999, and those choices are presented to the operator for human selection. Following this procedure, the word will either be correctly matched or among those choices presented more than 99.9% of the time.

This result depends, however, on the completeness of the reference dictionary. If the true match is not in the dictionary, the Bayesian function will still pick the *closest* match, and assign it a probability assuming that the correct match is indeed found in the dictionary. We have found in practice that scores produced under these circumstances are often as high as the correct matches, since the Bayesian score is actually a function of how much *better* than all the other matches a particular choice is, rather than simply how good it is when taken alone. These false positives are relatively few, but comprise the vast majority of the incorrect matches made by this algorithm. We found that we were able to detect most false positives by examining the numerator of Equation (10) (which is the probability of match function) along with the final quotient.

Ideally the false positive problem would be solved by generating an all-inclusive dictionary. It may seem counterintuitive that *increasing* the size of the reference dictionary (and thus increasing the number of possible mismatches) would *reduce* false positive results, but the use of the word frequency information in the dictionary allows this to happen. The word frequency information acts as a graded restriction on the dictionary, so that rather than words simply being present or absent in the dictionary, they are present at some frequency level. The algorithm simply assumes that any absent words have a frequency of zero. A collection of enough data over enough time will approximate this perfect dictionary, allowing the Bayesian scores to be meaningful in practical application.

4.4. Substitution matrix

Using a substitution matrix in the context of an edit distance calculation is often referred to as the 'alphabet-weighted edit distance' or simply 'weighted edit distance'. Various substitution matrices have been proposed for various schemes (they are used extensively in DNA and amino acid sequence searching where mutations are common), and the mathematics of the weighted edit distance calculation have been developed (partially at NLM by Altschul and Karlin at the National Center for Biotechnology Information[4,5]) with attention given to the optimum matrix for a given application and the interpretation of the match scores produced. The theory maintains that the optimum matrix is constructed from the substitution frequencies of the population that produced the sample under test.

A refinement of our substitution matrix is apparent. As they are implemented now, none of the ranking methods take advantage of the confidence data, which, as seen in the figures, contain valuable information that could be used in the match process. Presently, the substitution of a letter with confidence 9 carries no more penalty than the substitution of a letter with confidence 1, which is clearly not as it should be.

A substitution matrix constructed completely from the translations of characters with confidence 8 or 9 would be very different from one made with confidences of 1. In the case of the 8's and 9's, the diagonal elements would be much higher than the off-diagonal elements, because these confidences have accuracy rates that approach 100%, and there are very few substitutions. This is in contrast to the 1's, which have accuracy rates around 15%, and the diagonal elements would be much smaller in comparison to the rest of the matrix. Construction of separate matrices, one for each confidence level, would make optimum use of the both the confidence and frequency information available from thousands of previous records.

5. CONCLUSION

Five methods of matching OCR-mistranslated words with their original, true words were described. Of the five, the Bayesian approach, which takes into account the closeness of the match, the frequency of the true word in its vocabulary space, and the closeness and frequencies of all other words in the vocabulary space produced the most accurate results and had some benefits in interpreting the score in meaningful ways. This approach achieved an accuracy rate of 99.1% under the best conditions, and a rate of 85.7% under the strictest conditions. The other methods produced inferior but still excellent results, with the worst accuracy being 77.3 percent for the generic edit distance algorithm. Use of a probabilistic substitution matrix increased the accuracy from 77% to 85% over the generic edit distance.

REFERENCES

[1] http://www.ncbi.nlm.nih.gov/PubMed/

[2] Gusfield D. *Algorithms on Strings, Trees, and Sequences: computer science and computational biology*. Cambridge University Press, New York, 1997.

[3] Divita G, Browne A, Tse T, Cheh M, Loane R, and Abramson M. "A Spelling Suggestion Technique for Terminology Servers." Poster Session, AMIA Fall Symposium, Los Angeles, 2000.

[4] Altschul S. "Amino acid substitution matrices from an information theoretic perspective." *J Mol Biol*, **219,** 555-65, 1991.

[5] Karlin S, Altschul SF. "Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes." *Proc Natl Acad Science***, 87,** 2264-68, 1990.