EVALUATION OF APPROXIMATE PATTERN MATCHING ALGORITHMS FOR OCR TEXTS

Manolis Christodoulakis[1], Gerhard Brey[2] and Rizwan Ahmed Uppal[1]
[1]*School of Computing, Information Technology & Engineering, University of East London*
[2]*Centre for Computing in the Humanities, King's College London*
m.christodoulakis@uel.ac.uk, gerhard.brey@kcl.ac.uk, rauppal@hotmail.com

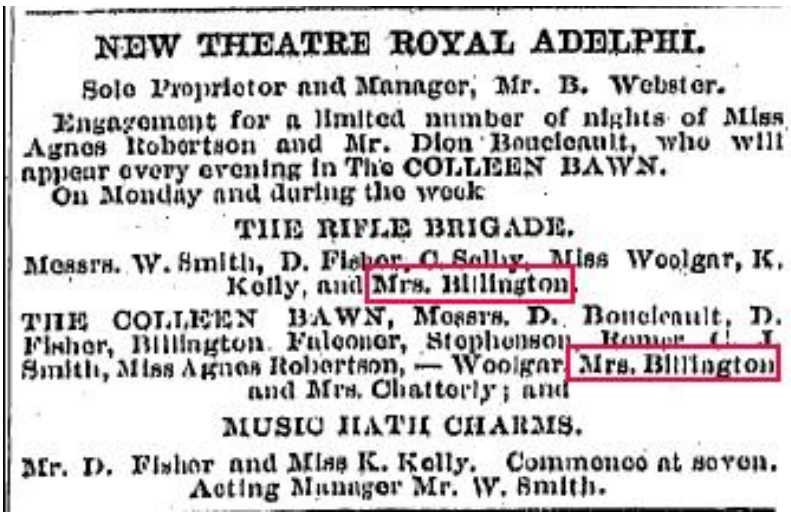
Abstract: In recent years there has been going on a large process of digitising old books, articles and newspapers. These documents are scanned and then processed with Optical Character Recognition (OCR) software to obtain their text equivalent. However, due to the (usually) poor quality of the original papers, the OCR software produces text which is not 100% accurate. A simple search for a pattern in the resulting text would only retrieve those occurrences that were accurately interpreted, but will ignore incorrectly spelled or distorted variations. In this paper we make use of the recently devised algorithm by Christodoulakis and Brey (2008), on the edit distance with combinations and splits, to perform approximate pattern matching for OCR texts. We then compare its performance against classic general-purpose approximate matching algorithms.

1. Introduction:

The process of "digitising" old books, articles, newspapers, etc. –i.e. the process of obtaining a digital representation of a document– is of immense importance to a large variety of groups of people, such as librarians, academics, publishers, etc. This task is achieved via Optical Character Recognition (OCR), where the scanned documents are interpreted to text, which can then be stored, searched for, indexed etc. However, when the original paper copies of the publications are of poor print quality, then also the quality of some of the text resulting from the OCR may vary from barely readable to illegible. That is, the resulting text contains errors, misspellings of the original text. A simple search of the scanned and processed text for a pattern would retrieve exact matches, but ignore incorrectly spelled or distorted variations.

Figure 1 demonstrates an example of the problem, from the Nineteenth-Century Serials Edition, NCSE in short, (NCSE, 2008), a digital edition of six nineteenth-century newspaper and periodical titles. The actress `Mrs. Billington` is mentioned twice. OCR recognised the name once as `Mrs. lUllinijton` and then as `Mrs. BIIMngton`. A simple search for the name `Billington` would therefore be unsuccessful. In fact, NCSE is full of documents of quality similar to or even worse than the one shown in Figure 1, due to the age of the original documents.

This problem is easier to overcome when the pattern is a common word, by checking the possibly misspelled matches of it against an authoritative electronic dictionary. But an equivalent reference work for names does not exist, while at the same time the most common types of queries in these old articles are names (of politicians, authors,



NEW THEATRE ROYAL ADELPHI.

Sole Proprietor and Manager, Mr. B. Webster.

Engagement for a limited number of nights of Miss Agnes Robertson and Mr. Dion Boucicault, who will appear every evening in The COLLEEN BAWN.

On Monday and during the week

THE RIFLE BRIGADE.

Messrs. W. Smith, D. Fisher, C. Selby, Miss Woolgar, K. Kelly, and Mrs. Billington.

THE COLLEEN BAWN, Messrs. D. Boucicault, D. Fisher, Billington, Falconer, Stephenson, Romer, C. J. Smith, Miss Agnes Robertson, — Woolgar, Mrs. Billington and Mrs. Chatterly; and

MUSIC HATH CHARMS.

Mr. D. Fisher and Miss K. Kelly. Commence at seven.

Acting Manager Mr. W. Smith.

Figure 1. Mispelled occurrences of Mrs. Billington

actors, etc.). Therefore, the algorithms we deal with in this paper are of great importance, especially for name searching, although they could of course be applied to any other type of pattern too.

Formally, in this problem one has to deal with locating "approximate" matches of a given pattern; that is, matches that are not exactly identical to the pattern, but rather are *similar* and are likely to correspond to the pattern. The notion of "approximation'' (or similarity) is measured through what is known as "distance metrics", which essentially measure the distance between two strings; the larger the distance the less similar the strings are.

In different contexts, string similarity (or string distance) is defined in different ways and there exist today numerous special-purpose and general-purpose distance metrics, see for example (Navarro, 2001; Crochemore & Rytter, 2002; Smyth, 2003). One of the most common is the "Edit Distance" (Levenshtein, 1966), a general-purpose metric which defines the distance between two strings as the minimum number of character insertions, deletions and substitutions that are required to transform one of the strings under examination to the other. Another commonly used general-purpose distance metric is the *n-gram count* (Cavnar & Trenkle, 1994), where similarity of two strings is measured in terms of their common substrings of length *n*.

Very recently, Christodoulakis and Brey (2008) presented a new distance metric that is specifically tailored to OCR texts. In particular, their metric, called "Edit Distance with Combinations and Splits" (abbreviated as EDSSC), defines the distance between two strings by taking into account, not only insertions, deletions and substitutions, but also optical similarities of characters and combinations of characters. For example, this new metric incorporates the fact that the character "m" resembles the combination of characters "rn" ("r" followed by "n"), and would thus assign a small distance between these two.

In this paper, we use the EDSSC algorithm to perform pattern matching on OCR texts, and we evaluate its performance against the approximate pattern matching algorithm that uses the simple edit distance or the n-gram count distance. The main focus of the evaluation is the accuracy of the resulting matches (precision and recall) with respect to the maximum distance allowance.

The paper is organised as follows. In Section 2 we describe the three distance metrics that we evaluate in this paper. In Section 3 we briefly describe how the EDSSC algorithm works and in Section 4 our approximate pattern matching algorithm where the pattern is a name. In Section 5 we present our experimental results and our evaluations. Section 6 contains our concluding remarks.

2. Distance Metrics:

Consider strings $x=x[1]...x[n]$ and $y=y[1]...y[m]$ over an alphabet Σ; the edit distance between x and y is the minimum number of *edit operations* (insertions, deletions or substitutions) to transform x to y, or vice versa (Sankoff & Kruskal, 1983; Levenshtein, 1966). Implicitly, the simple edit distance assigns to each operation a unit cost, and returns the minimum cost of transforming x to y as the distance between x and y. A generalised version of the edit distance, is one that allows the different operations to have different costs; let d_{sub} be the cost for one substitution operation, and d_{indel} that of one insertion or deletion operation (one is a dual of the other, hence the identical cost). The generalised edit distance is defined then as the minimum cost of transforming x to y.

In the variant of the edit distance introduced in (Christodoulakis & Brey, 2008), during the comparison of the two strings x and y, a character α in x may match any of the following in y:

- an occurrence of α in y (i.e. α matches itself);
- a different character β that looks similar to α;
- a combination of characters $\beta_1\beta_2...\beta_k$ which as a whole looks similar to α.

The cost of the first operation is of course 0. The cost of the other two operations can also be set to 0 (since these operations denote "matches"), but for the sake of accuracy we prefer to set it to a predefined constant d_{comb}, much smaller than d_{sub} or d_{indel}.

For example, the character "*m*" could match, apart from itself, the strings "*in*", "*rri*", "*iri*" and many more combinations that optically resemble "*m*"; similarly, "*a*" could match "*o*", "*ci*", etc. The list of all the valid matches for a given character α is called the *combination list* of α and is formally defined as

C_α={all the combinations $\beta_1\beta_2...\beta_k$, for $k\geq 1$, such that α resembles $\beta_1\beta_2...\beta_k$}

We then call α a *valid combination* of $\beta_1\beta_2...\beta_k$, and equivalently $\beta_1\beta_2...\beta_k$ is a *valid split* of α.

The third distance metric we evaluate is the general-purpose *n-gram count* approach (Cavnar & Trenkle, 1994). In this method, the strings under consideration are broken down to (overlapping) substrings of length n. For instance, the 2-grams of the word "hello" would be "he", "el", "ll", "lo". Then the number of common n-grams between the two strings denotes their similarity.

3. The EDSSC Algorithm:

The algorithm for computing the edit distance with combinations and splits (EDSSC algorithm) operates in two stages.

In the first stage, it pre-processes the lists of valid combinations for all the characters. It constructs one Aho-Corasick automaton (slightly modified for the purposes of this algorithm) for each combination list C_α. Figure 2 demonstrates the Aho-Corasick automaton for the sample combination list

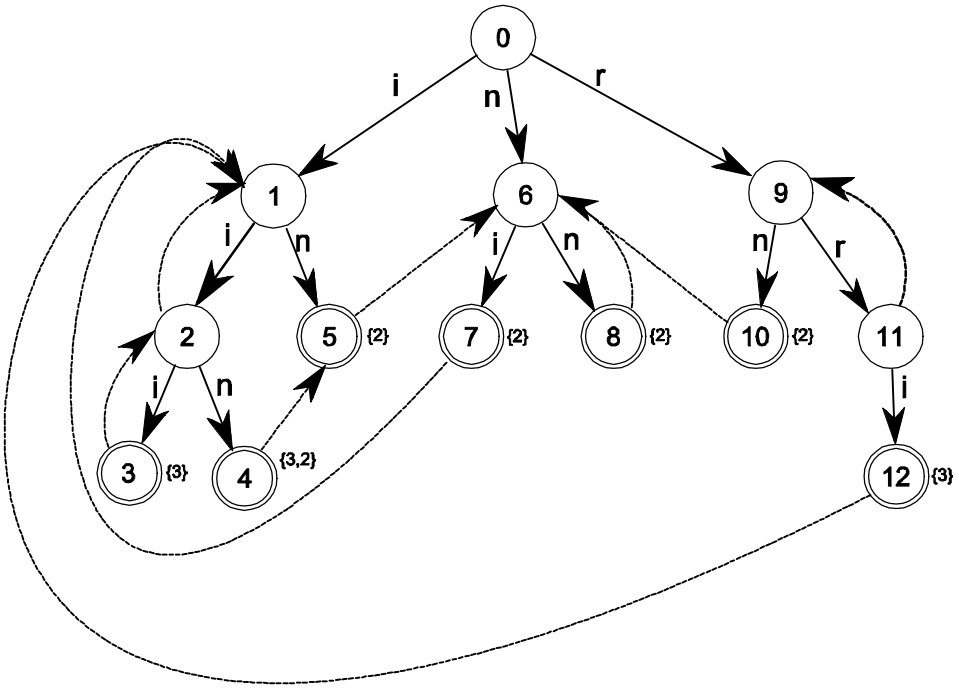


Figure 2. Aho-Corasick automaton for the combination list C_m

C_m={*iii, iin, in, ni, nn, rn, rri*}

Obviously, this stage of the algorithm, is only computed once, at the very beginning, to load all the valid combinations. The running time of processing is proportional to the sum of the lengths of all the valid combinations.

The second stage of EDSSC, computes the actual distance (or cost) between the two strings *x* and *y*, by using a form of dynamic programming (Cormen, et al., 2001), enhanced with combination and split operations. For two strings of length *m* and *n* respectively, this stage operates in time $O(mnc)$, where *c* is an upper-bound on the number of valid splits for any character α. It is this part of the algorithm that we run repeatedly in the pattern matching algorithm presented below.

4. Approximate Name Matching:

As mentioned earlier, the focus of our evaluation is on name matching; names are used much more often as search terms in NCSE and other similar databases of articles and periodicals. Luckily, NCSE has already pre-processed its texts and extracted huge lists of names (together with links to the documents where they appear), the largest majority of which are misspelled. The algorithm we present in this section operates on these extracted lists of names.

Our algorithm is a simple heuristic one that searches for a given name, the *pattern*, in this large list of misspelled names, the *text*, and returns the occurrences that look similar to the pattern. But since we are looking for *approximate* occurrences (called *approximate matches*) that are not identical to the pattern, we must also specify in the input the maximum "distance" that a name in the list may have from the pattern to be

considered a match; this distance is given as a percentage and has a different meaning for each of the distance metrics that we tested. As one would expect, there is a trade-off in distance-quality of matches: low distance threshold, yields less false positives, but may miss some true matches; on the other hand, a high distance threshold has less chances of missing true matches, but will return many fake ones too.

The algorithm consists of two main components. First, the algorithm finds the distance between the pattern and every name in the text whose distance from the pattern has not already been computed. The distance is the edit distance, the EDSSC or the n-gram count, depending on the user's preference. For the purposes of our evaluation we run all queries for all three distance metrics and compare the results (see Section 5). For every name in the text, the algorithm checks whether its distance is less than the threshold distance provided by the user; if it is not, then name is not considered a "match", i.e. it is not similar to the pattern, and thus it is discarded.

However, identifying all the matches is not enough since, depending on the maximum allowed distance, their number might be quite large. For this reason, it is important to use a ranking system, to sort the matches and bring higher in the results the better ones, the ones that are more likely to correspond to the pattern. The rank of a match depends on:

1. its distance from the pattern; the smaller the distance the higher the rank, since smaller distance signifies larger similarity to the pattern;
2. the number of occurrences of the particular match; the larger the number of occurrences the better, since it is common for the OCR software to repeatedly misinterpret the same characters (e.g. many occurrences of "Palrnerston" would correspond to "Palmerston" –"r" followed by "n", instead of "m")
3. whether the match is a valid name on not; if it is, then the rank will get lower, assuming that the pattern is not misspelled. Consider for example the pattern "Billington" (a valid name) and the approximate match "Wellington"; their distance is small (two substitutions), but if one notices that the latter is itself a valid name, he may conclude that in reality it is not a match of the pattern; it is more likely an occurrence of the pattern "Wellington" rather than a mis-spelled occurrence of "Billington". The validity of a name is verified by looking it up in a prespecified list of valid names. Of course, this list may never be considered complete, and thus the validity of matches should only play a small role in their rank.

From the above three criteria that ranking depends on, most important is the distance; the other two are heuristics that we have added in the algorithm, which from our experience further help bring good matches higher up.

5. Name Matching Evaluation:

In this section we present our experiments with the approximate name matching algorithm under three different distance metrics. The purpose of the experiments was to compare how the different distance metrics perform in the context of name matching, where the data originated from scanned and OCR'd texts. In particular, for each of the queries that we tested for approximate name matching, we conducted three variants of the name matching algorithm. The variants only differed from

each other in the distance metric in use; the first one was using simple edit distance, the second n-grams, and the third edit distance with combinations and splits.

The algorithms were implemented in C++, using the STL library. All tests were conducted on a 2GHz Intel Core 2 Duo processor with 2GB main memory. The operating system was Microsoft Windows Vista, Business edition. The compiler used was the GNU gcc, version 3.4.5 (mingw-vista special). The dataset we run the experiments on originated from the NCSE and consisted of a huge list of misspelled names (523,687 names), one name per line followed by the filename (and path) of the file that contained the specific name.

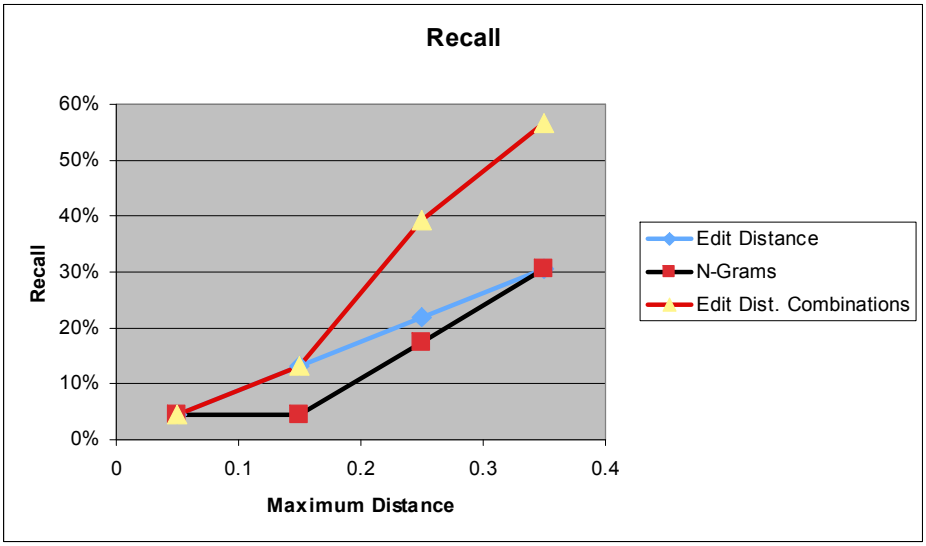


Figure 3. "Pseudo-recall" for the three distance metrics, for pattern "Billington"

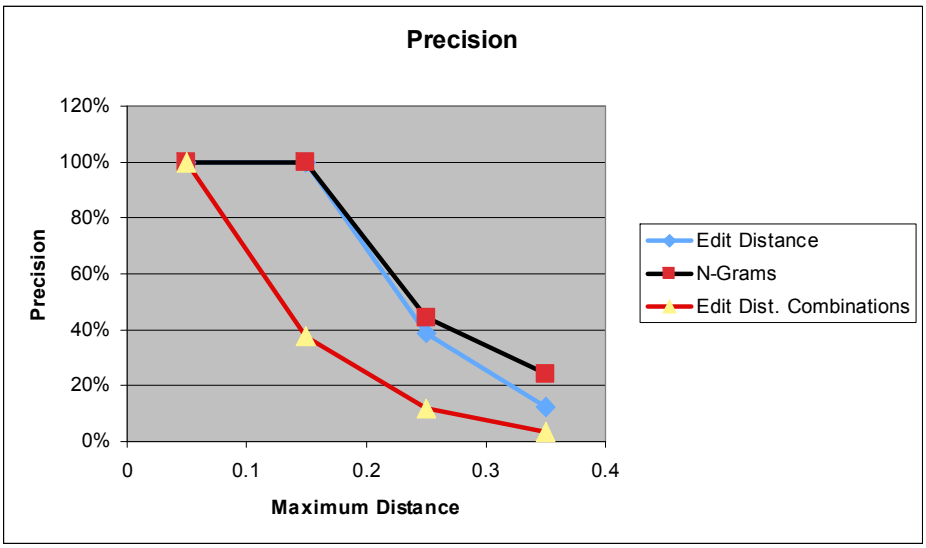


Figure 4. "Pseudo-precision" for the three distance metrics, for pattern "Billington"

In order to measure the effectiveness of the different distance measures, one needs to

measure the recall of the algorithms (that is, the percentage of the correct matches that the algorithm identified) and the precision of the algorithms (that is, the percentage of the identified possible matches that are true matches). However, in a database of the size of NCSE (with approximately 100,000 articles), it is not possible to manually scan all the documents and identify the correct spelling of all the misspelled words; if it were, there would not be any need for an approximate matching algorithm. Consequently, it is not possible to properly measure the recall and precision of any algorithm that operates on this set of data.

For this reason, we identified a number of approximate matches of the pattern that was used in this experiment, "Billington", and manually scanned the documents to identify whether these misspelled matches really corresponded to occurrences of "Billington". In this way, we could identify a "pseudo-recall" of the algorithms in the sense that we could answer the question "what percentage of the *known* matches has the algorithm identified"; similarly, we computed "pseudo-precisions" by answering the question "what percentage of matches that the algorithm identified belong to the *known* true matches?". These two pseudo-measurements are of course not providing the full picture, but are still very useful in giving us an idea of how well the algorithms are performing in comparison to each other.

Figures 3 and 4 demonstrate the efficiency of the algorithm for the three different distance metrics. The *x*-axis is the maximum distance that the pattern was allowed to have from a candidate match for it to be considered a match. In Figure 3 we can see that, even for a maximum distance as high as 35%, the recall is only about 30% for both the simple edit distance and the n-gram distance, with the latter performing slightly better; on the other hand, when the EDSSC distance was in use, the recall ratio almost doubles (57%). Figure 4 demonstrates that the precision rate of EDSSC is significantly worse than the corresponding rate for the other two distance metrics for small threshold distances, but they all tend to converge for distance larger than 35%. The low precision rate of the EDSSC came as no surprise, since the combinations and splits that are now allowed at very small cost imply a much larger number of possible approximate matches.

We performed similar experiments with other patterns too, with similar or often even better results. An interesting example was the pattern "Palmerston", which appears to have possibly more than one thousand different spellings (we verified 715 true matches that had been misspelled, by only looking at a sample of the documents). In this case, all three algorithms performed well, but the pseudo-recall rates of EDSSC were significantly higher than that of the other two distance metrics, while the pseudo-precision was only slightly smaller.

6. Conclusions:

This paper focused on the problem of approximate pattern matching for texts that have been produced through document scanning and OCR processing. We evaluated the effectiveness of three distance metrics, the "edit distance" the "n-gram count" and the "edit distance with combinations and splits" (EDSSC) on bad-quality OCR'd texts, by performing approximate name matching.

Our experiments indicate that the recently developed EDSSC distance metric is a promising tool for such applications, but there also appears to be room for improvement. More research needs to be

conducted to achieve better precision and possibly further increase the recall rate too.
But approximate string matching is only one way to tackle this problem; an interesting alternative research direction could be machine learning techniques, for example, so that the lists of valid combinations are created and adapted dynamically, depending on the input texts, rather than be statically hard-coded in the program.

7. References:

Christodoulakis M., Brey G., "Edit Distance with Single-Symbol Combinations and Splits", In, Jan Holub and Jan Zdarek, editors, *Proceedings of the Prague Stringology Conference (PSC)*, Prague, Czech Republic, September 2008, pp. 208-217.

Cormen T., Leiserson C., Rivest R., Stein C., *Introduction to Algorithms*, 2nd ed. MIT Press & McGraw-Hill, 2001, pp. 323-69.

Crochemore M., Rytter W., *Jewels of Stringology: Text Algorithms*, World Scientific, 2002.

Levenshtein V., "Binary codes capable of correcting deletions, insertions and reversals", *Soviet Physics Doklady*, 1966, pp. 707-710.

Navarro G., "A guided tour to approximate string matching", *ACM Computing Surveys*, ACM, New York, USA, 2001, pp. 31-88.

Nineteenth-Century Serials Edition (NCSE). http://www.ncse.ac.uk, 2008

Sankoff D., Kruskal J., *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, Addison-Wesley, 1983.
Smyth B., *Computing Patterns in Strings*, Pearson Education, 2003.

Cavnar W., Trenkle, J., "N-Gram-Based Text Categorization", In *Proceedings of SDAIR-94, 3rd Annual Symposiumon Document Analysis and Information Retrieval*, Las Vegas, 1994, pp. 161–175.