Approximate String Matching for Detecting Keywords in Scanned Business Documents

Thi Hien Ha

Natural Language Processing Centre
Faculty of Informatics, Masaryk University
Botanická 68a 602 00 Brno, Czech Republic
462259@mail.muni.cz

Abstract. Optical Character Recognition (OCR) is achieving higher accuracy. However, to decrease error rate down to zero is still a human desire. This paper presents an approximate string matching method using weighted edit distance for searching keywords in OCR-ed business documents. The evaluation on a Czech invoice dataset shows that the method can detect a significant part of erroneous keywords.

Keywords: approximate string matching, Levenshtein distance, weighted edit distance, OCR, invoice

1 Introduction

Business documents, different from other types of documents, are obligation to have a predefined set of data which is usually specified by keywords. Therefore, localization of keywords in the document plays an important role in document processing. However, scanned business documents potentially involve OCR errors which exact match cannot solve.

A deep statistical analysis of OCR errors covering five aspects, based on four different English document collections was described in [7]. They find out that among three most common edit operation types in OCR errors (insertion, deletion, and substitution), substitution is much more frequent than the others with average of 51.6%, and most of OCR errors can be corrected just by single operation types (total percentage of three single operations is 77.02%). The analysis also results in detailed statistics of standard mapping and non standard mapping, which is valuable to create character confusion matrix, one of the most important sources for generating and ranking candidates in error correction.

OCR post-processing aiming at fixing the residual errors in OCR-ed text. The approaches for this problem can be categorized into dictionary-based and context-based types. Dictionary and character n-gram are often used in the former to detect and correct isolated word errors whereas the latter consider grammatical and semantics context of the errors. This usually relies on word n-grams and language modeling [3,2].

However, business documents such as invoices has different characteristics in comparison with data using in those methods. Firstly, invoices are written

A. Horák, P. Rychlý, A. Rambousek (eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2019, pp. 49–54, 2019. © Tribun EU 2019

in short chunk instead of fully grammatical text. Secondly, non-words are frequent in invoices, involving entities names, and almost all of data fields such as invoice/order number, item codes, account number. Approximate string matching is string matching of a pattern in a text that allows errors in one or both of them [6]. It very soon became a basic tool for correcting misspelling words in written text, and then, in text retrieval since exact string matching was not enough due to large text collection, more heterogeneous, and more error-prone.

In this paper, we use approximate string matching based on a weighted edit distance to search for keywords in scanned business documents. The method is evaluated on a Czech invoice dataset.

2 Method

2.1 Problem definition

The problem of approximate string matching is defined as follow:
Let Σ be a finite alphabet; $|\Sigma| = \sigma$; Σ^* is the set of all strings over Σ
Let $T \in \Sigma^*$ be a text of length n; $|T| = n$
Let $P \in \Sigma^*$ be a pattern of length m; $|P| = m$
Let d: $\Sigma^* \times \Sigma^* \rightarrow \Re$ be a distance function.
Let $k \in \Re$ be the maximum number of error allowed.
The problem is given T, P, k and d(.), return the set of all the substrings of T: $T_{i..j}$ such that $d(T_{i..j}, P) \leq$ k.

The distance d(x, y) between two strings x and y is the minimal cost of a sequence of operations that transform x into y. The operations are a finite set of rules ($\delta(z, w)$). In most of applications, the set of operations is limited to:

- *Insertion*: $\delta(\varepsilon, a)$: inserting the letter a
- *Deletion*: $\delta(a, \varepsilon)$: deleting the letter a
- *Substitution*: $\delta(a, b); a \neq b$: substituting a by b
- *Transposition*: $\delta(ab, ba); a \neq b$: swap the adjacent letters a and b.

One of the most commonly used distance function is Levenshtein [1], also called edit distance. Edit distance $d(x, y)$ between two strings x, y is the minimal number of insertions, deletions, and substitutions to transform x into y. The distance is symmetric, i.e. $d(x, y) = d(y, x)$. In the simplified definition, all the operations cost 1. Therefore, $0 \leq d(x, y) \leq max(|x|, |y|)$.

2.2 Weighted edit distance

Detecting keywords from OCRed documents faces at least two problems. The first problem to take into account is OCR errors. There are both standard mapping 1:1 (one character is mis-recognized into another character, e.g "email" and "emall") and non-standard mapping 1:n or n: 1 (e.g "rn" and "m"). However, the dominant portion of OCR errors is standard mapping. More example of common errors can be seen in Table 1. Another type of OCR errors is incorrect word boundary, including incorrect split (i.e wrongly splitting one word into two or more strings) and run-on error (i.e inaccurately putting two or more words together).

Table 1: Common pairs of OCR errors

Character 1	Character 2	Character 1	Character 2
b	h	n	r
c	o	O	0
c	e	r	i
C	(	r	t
f	t	s	5
f	l	v	y
i	l	z	2
l	|	z	s
l	1	y	g
l	ť	m	n

Beside OCR errors, the other problem comes from the language characteristics. Modern Czech orthographic system is diacritical. The acute accent and *háček* are added to Latin letters, such as "á", "í", "ě", "č". Moreover, grammatically, Czech is inflectional, like other Slavic languages. The missing or excessive diacritics, using different endings, either by typing errors or OCR errors make the problem worse.

Being aware of those problems, we set different costs for operations. For those substitutions of common OCR errors, or substitutions between pairs of short and long vowels, with or without háček (from now on, they are mentioned as common OCR errors), we set a much cheaper cost (e.g 0.1) than the normal one (i.e 1). The lower cost is also set for inserting or deleting of spaces and punctuation. We call this as a weighted edit distance function.

$$\delta(a,b) = \begin{cases} 0 & \text{if } a = b \\ 0.1 & \text{if } \left[\begin{array}{l} (a = \varepsilon \text{ and b is punctuation}) \\ \text{or } (b = \varepsilon \text{ and a is punctuation}) \\ \text{or } (a,b) \text{ is a common pair of OCR errors} \end{array}\right. \\ 1 & \text{otherwise} \end{cases}$$

Because insertion and deletion have the same weight, the distance function is still symmetric.

2.3 Algorithms

We call $\alpha = k/m$ the error ratio. Since we can make the pattern match at any position in the text by performing m substitutions, $0 \leq k \leq m$ (reminding $m = |P|$). Therefore, $0 \leq \alpha \leq 1$.

In the problem of searching for keywords in the text allowing an error ratio α, P is the keyword. We propose a filtering algorithm involving following steps:

- Get the longest consecutive common substring of the keyword and the text. If the ratio between length of the substring and length of the keyword is less than a third, then return no approximate match found.
- Extend the substring in the text to get the longest substring with smallest edit distance.
- If the ratio between weighted edit distance of the keyword and new substring and length of the keyword does not exceed α, then return the substring. Otherwise, return no approximate match found.

Weighted edit distance function using dynamic programming:

```
function Weighted_distance(s1[0..m-1], s2[0..n-1]):
  float prev_row[0..n]
  float  cur_row[0..n]
  prev_row[0] = 0
  for i from 1 to n:
      prev_row[i] = prev_row[0] + δ(ε,s2[i-1])
  for i from 1 to m:
      cur_row[0] = prev_row[0] + δ(s1[i-1],ε)
      for j from 1 to n:
          insertions = prev_row[j] + δ(ε, s2[j-1])
          deletions  = cur_row[j-1] + δ(s1[i-1], ε)
          substitutions = prev_row[j-1] +
                          δ(s1[i-1], s2[j-1])
          cur_row[j] = min(insertions, deletions,
                                      substitutions)
      prev_row = cur_row
  return prev_row[-1]
```

For Levenshtein or edit distance function, we just need to replace δ function by 1. Because weighted edit distance function is slower than normal edit distance, in the the second step, we use original edit distance and only calculate weighted edit distance at the final step.

There has been a significant number of research in implementation of approximate string matching to reduce time and space complexities. Methods based on finite state automation promise to be much faster than dynamic programming and can be computed in linear time [8,4,5].

3 Experiments

The dataset contains 50 Czech scanned invoices. After using an open source OCR to get the text, we run two different modules to detect a set of given keywords. The former is exact matching using regular expression. The latter uses proposed

Table 2: Analysis of kewords detected by approximate string matching but missed by exact matching

Types of error	Number of keywords	in %
OCR errors	52	30.4%
Diacritics	32	18.7%
Inflection	44	25.7%
False possitive	43	25.1%
Total	171	100%

approximate string matching. Then, we compare to see how many keywords the approximate string matching detected that exact matching did not.

Keywords for 36 fields are detected, including invoice number, invoice date, order number, order date, due date, payment date, and so on. The length of keywords varies from 1 (e.g invoice number: "č") to 43 (e.g payment date: "datum uskutečnění zdanitelného plnění").

The threshold for the error rate is set to 0.15 in the experiment. This means, for example with a keyword of length 6, the distance allowed is 6×0.15 = 0.9, i.e it allows only common OCR errors.

The result is summarized in Table 2. In total 171 keywords detected by approximate string matching but missing by exact match, 30.4% are because of OCR errors in one characters (e.g "C′Slo" instead of "Cislo", "Odběrate!" instead of "Odběratel", "e-mall" instead of "e-mail"), 18.7% caused by missing or excess of diacritics (e.g in "č", "ě", "í"). The different endings by inflection, for instance "objednávka" and "objednávky", are the reason for 25.7%. The last 25.1% are caused by close keywords. Let take the title "invoice" and the field "invoice number" as an example. One of keywords for this field is "Daňový doklad č" which has only a space and one character ("č", abbreviation of "číslo", means "number") differ from the title "Daňový doklad". Therefore, the distance (d = 1.1/15≈0.07) is less than the threshold (0.15), resulting title is marked as keyword. This error can be filtered out by checking if the annotated substring is also marked as title. In fact, in many invoices, there is an invoice number on the same line of title without the word "number" accompanied. Therefore, title in these cases is the signpost to extract the invoice number, the same role as a keyword.

Besides, we notice that there are 20 keywords containing OCR errors are still missing by approximate string matching. These errors are not in the given OCR common errors. Almost half of them are standard mapping, e.g "o" becomes "g", "I" becomes "′", or "ft" becomes "||". The other half are caused by non standard mapping, such as "čt" becomes "d", or "j" becomes "f′".

4 Conclusion

In this paper, we have described approximate string matching approach based on a weighted edit distance for detecting keywords in scanned business documents.

The experiment shows that this method adapts pretty well to erroneous text and inflectional languages. The future work will focus on a complete list of common errors and implementation using finite state automation.

Acknowledgements This work has been partly supported by Konica Minolta Business Solution Czech within the OCR Miner project. This publication was written with the support of the Specific University Research provided by the Ministry of Education, Youth and Sports of the Czech Republic.

References

1. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet physics doklady. vol. 10, pp. 707–710 (1966)
2. Llobet, R., Cerdan-Navarro, J., Perez-Cortes, J., Arlandis, J.: Ocr post-processing using weighted finite-state transducers. In: 2010 20th International Conference on Pattern Recognition. pp. 2021–2024 (Aug 2010). https://doi.org/10.1109/ICPR.2010.498
3. Mei, J., Islam, A., Moh'd, A., Wu, Y., Milios, E.: Statistical learning for ocr error correction. Information Processing & Management **54**(6), 874–887 (2018)
4. Mihov, S., Schulz, K.U.: Fast approximate search in large dictionaries. Computational Linguistics **30**(4), 451–477 (2004)
5. Mitankin, P.: Universal levenshtein automata. building and properties. Sofia University St. Kliment Ohridski (2005)
6. Navarro, G.: A guided tour to approximate string matching. ACM computing surveys (CSUR) **33**(1), 31–88 (2001)
7. Nguyen, T., Jatowt, A., Coustaty, M., Nguyen, N., Doucet, A.: Deep statistical analysis of ocr errors for effective post-ocr processing. In: 2019 ACM/IEEE Joint Conference on Digital Libraries (JCDL). pp. 29–38 (June 2019). https://doi.org/10.1109/JCDL.2019.00015
8. Schulz, K.U., Mihov, S.: Fast string correction with levenshtein automata. International Journal on Document Analysis and Recognition **5**(1), 67–85 (Nov 2002). https://doi.org/10.1007/s10032-002-0082-8, `https://doi.org/10.1007/s10032-002-0082-8`