УДК 519.686

Simple Essential Improvements to the ROUGE-W Algorithm

Sergej V. Znamenskij*
Ailamazyan Program Systems Institute of RAS
Peter the First Street, 4, Veskovo village
Pereslavl area, Yaroslavl region, 152021
Russia

Received 10.10.2015, received in revised form 01.11.2015, accepted 16.11.2015

The ROUGE-W algorithm to calculate the similarity of texts is referred in more than 500 scientific publications since 2004. The power of the algorithm depends on the weight function choice. An optimal selection of the weight function is studied. The weight functions used previously are far from optimality. An example of incorrect output of the algorithm is provided. Simple changes are described to ensure the expected result.

Keywords: sequence alignment, longest common subsequence, ROUGE-W, edit distance, string similarity, optimization, complexity bounds.
DOI: 10.17516/1997-1397-2015-8-4-497-501

1. The ROUGE-W problem

Let Σ be an alphabet set. We note a starting segment of natural numbers set $\mathbb{N}$ as $\overline{1,n} = (1,\dots,n)$. *String* is the finite sequence of letters $X = (x_1,\dots,x_n)$ that can be considered as a function $X\colon \overline{1,n} \to \Sigma$ returning a letter located at the given position.

For two given strings $X = (x_1,\dots,x_m)$ and $Y = (y_1,\dots,y_n)$ a *common subsequence* P of X and Y with a length $k \leqslant \min(m,n)$ is usually defined as a couple of finite sequences $P_X = (p_{X1}, p_{X2},\dots,p_{Xk})$ and $P_Y = (p_{Y1}, p_{Y2},\dots,p_{Yk})$ meeting the equation

$$X(p_{Xi}) = Y(p_{Yi})\,. \tag{1}$$

The common subsequence is called a *common substring* if $P_{Xi} = P_{X1}+i-1$ and $P_{Yi} = P_{Y1}+i-1$ for all $i \geqslant k$. The well known sequence alignment problem is to find the most valuable common subsequence for any given strings. A common subsequence P became the Longest Common Subsequence (LCS) if "the most valuable" means "with the maximal possible length k". Since 70^{th} it is well known that such meaning does not meet practical needs [1]: when alignment intended to identify common part and difference in computer logs, LCS often finds unnaturally fragmented common part bearing a lot of frequently used letters being sporadically aligned.

For example, the option ~~pbe a refversenced being revfersenced~~ has the common sequence of the length equal to 11 against of only 10 symbols observed for be a reversed preference ~~being reversed~~ that is much better for text editing. A lot of other applications (partially mentioned in [2]) make reason to consider a long string or a chain of close longer strings to be preferably found in a common part than just a long list of very short senseless matches as in this example of edit distance for texts.

Comparison of string (A) to strings (B) and (C) shown below

(A) preference being reversed,

*svz@latex.pereslavl.ru

(B) be a reversed preference,
(C) a pure repared refresher,

yields the LCS (A, B)="`reerereere`" to be much shorter then LCS (A, C) = "`a reered refreer`" thought C looks less similar to A.

An idea to pay more attention to longer substring alignment made a progress. The common substrings of the length multiple to a fixed k are used in [3] and some later works for a "more accurate" definition of analogies to LCS and Levenshtein metric. Another point is that LCS and relative techniques such as Levenshtein distance have insufficient resolution for nice subsequence selection [4]: score range $\overline{1,n}$ is probably too poor for proper selection from exponentially huge amount of possible alignments.

The first carefully described similarity evaluation tool, paying more attention both to longer substrings and implementation of larger scale for better selection, appeared in [5] called FLCS. Yet, several years earlier more general tool has been introduced in very popular framework ROUGE [6, 7].

The ROUGE-W algorithm seeks for the common substrings $P_i = (P_{iX}, P_{iY})$ that compose the joint subsequence

$$\begin{aligned} P_X &= (P_{1X1}, \ldots, P_{1Xl_1}, P_{2X1}, \ldots, P_{2Xl_2}, \ldots, P_{pX1}, \ldots, P_{pXl_p}) \\ P_Y &= (P_{1Y1}, \ldots, P_{1Yl_1}, P_{2Y1}, \ldots, P_{2Yl_2}, \ldots, P_{pY1}, \ldots, P_{pYl_p}) \,, \end{aligned} \tag{2}$$

where $P_{iX} = (P_{iX1}, \ldots, P_{iXl_i})$ and $P_{iY} = (P_{iY1}, \ldots, P_{iYl_i})$ and the solutions of the following problem:

Problem 1 (ROUGE-W)**.** *For two given strings X, Y of the length m and n, correspondingly, and given positive convex increasing function f find a maximal over all common subsequences value $W(X, Y) = \max\limits_{P} W_f(P)$ of the weight*

$$W_f(P) = \sum_{i=1}^{p} f(l_i) \,, \tag{3}$$

where P consists of p common substrings P_i of the length l_i.

This problem was originally known as WLCS (weighted LCS), but later [8] while later this name has been related to another generalization of LCS.

The efficiency of algorithm heavily depends on the choice of weight function f which is a special problem. Linear $f(k) = \alpha k - \beta$ with $\alpha, \beta > 0$ and $f(k) = k^\alpha$ with $\alpha = 1{,}3$ was reported previously in a number of publications, while no reasonable recommendation for the choice of function f in ROUGE-W algorithm has been published so far. Authors of the ROUGE-W framework use $f(k) = k^\alpha$ with $\alpha = 2$ in the text of paper.

2. The choice of weight function optimization problem

Consider the simplest model for the function of evaluation from [9]. The algorithm should properly detect any common substring whit may have special meaning (the word, the meaningful part of word, key phrase etc.) So we consider start and length of possible random common substring S to be uniformly distributed (with no relation to (1) in the distribution for simplicity) and consider two alignments P and P' to be equivalent ($P \sim P'$), if S is found in them with the same probability. Then the objective functional

$$F(f) = \sum_{P \sim P'} \big(W_f(P) - W_f(P')\big)^2 \tag{4}$$

measures unrelated variation value of the weight function f.

Theorem 2.1. *A weight function* $f(k) = \dfrac{k(k+1)}{2}$ *yields a minimal value for the functional* F.

Proof. Since $F(f) \geqslant 0$, so far any existing solution of $F(f) = 0$ equation is optimal. Note

$$p(S, P) = \text{the probability for } S \text{ to be found in } P,$$

$$p_0 = \text{the probability of } S \text{ to coincide with a given common substring and } s(m, n), \text{ and}$$

$$\|P\| = \frac{p(s, P)}{p_0} \text{ the total number of common substrings in } P.$$

For $f(k) = \dfrac{k(k+1)}{2}$ we have $W_{f,P} = \|P\|$ that proves (4). □

There is a simple explanation of this optimal weight: it makes $W_{f,P}$ equal to the total number of common substrings in P.

Each substring may be shown with an angle pointing to its end; Fig. 1 illustrate the case $W(A, B) = 55$ and $W(A, C) = 15$, that is much more realistic than LCS values 10 and 16. The scores for $f(k) = k^2$ in this case also look nice: the figures are 100 and 19, respectively.

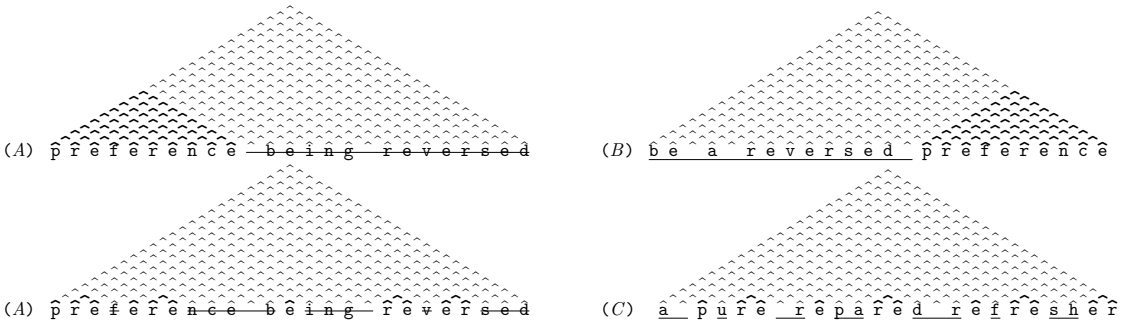

Fig. 1. Comparison of subsequences with the optimal weight function

3. On algorithm complexity of the algorithm

The original ROUGE-W algorithm does not always perform as expected. For example, one can expect optimal W (`"visitor is sit to or"` , "`elegance visitor`") $= f(7)$ due to common substring "`visitor`". While the original algorithm from [6, 7] finds the sequence consisting of `"v"`, `"i"`, `"si"`, `"t"` and `"or"` with much smaller score $3f(1) + 2f(2)$. The algorithm produces 11 against 49 for $f(k) = k^2$ weight, 9 against 28 for optimal weight and 2 against 6 for $f(k) = k - 1$. Only the LCS $f(k) = k$ gives the same 7.

For partial case $f(k) = k^\gamma$ the two correct algorithms were briefly described in [5] and for optimal selection of f in [10]. The dynamic programming versions are essentially the same in both papers an can be easily extended to the following simple generic algorithm:

```
for (i = 0; i <= m; i++){
    c[i,0] = 0; // initialize totals table
    c[0,i] = 0; // for left upper rectangles
for (i = 1; i <= m; i++){
```

```
    for (j = 1; j <= n; j++){
        c[i,j] = max (c[i-1,j],c[i,j-1]);
        b[i,j]=c[i,j];
        k = 0;
        while ( x[i-k] == y[j-k]){
            if(c[i,j] < b[i-k,j-k] +f(k+1)){
               c[i,j] = b[i-k,j-k] +f(k+1))}
            k=k+1;}}}
return c[m,n]
```

The internal cycle rarely runs two or more times and does not increase the execution time for short strings. Effective work with huge data will probably need the optimised version from [5] to be adapted.

Acknowledgments

This work was performed under financial support from the Government, represented by the Ministry of Education and Science of the Russian Federation (Project ID FMEFI60414X0138); also it was partly supported by a research grant No. 14.Y26.31.0004 from the Government of the Russian Federation.

References

[1] P.Heckel, A technique for isolating differences between files, *Commun. ACM*, **21**(1978), no. 4, 264–268.

[2] S.V Znamenskij, A Belief Framework for Similarity Evaluation of Textual or Structured Data, *Similarity Search and Applications*, **LNCS 9371**(2015), 138–149.

[3] G.Benson, A.Levy, R.Shalom, Longest Common Subsequence in k-length substrings, **arXiv:1402.2097**, 2014.

[4] K.-T.Tseng, C.-B.Yang, K.-S.Huang, The better alignment among output alignments, *Journal of Computers*, **3**(2007), 51–62.

[5] Y.-P.Guo, Y.-H.Peng, C.-B.Yang, Efficient algorithms for the flexible longest common subsequence problem, Proceedings of the 31st Workshop on Combinatorial Mathematics and Computation Theory, 2014, 1–8.

[6] C.Y.Lin, Rouge: A package for automatic evaluation of summaries, Text summarization branches out: Proceedings of the ACL-04 Workshop, **8**(2004).

[7] C.-Y.Lin, F.J.Och, ORANGE: a Method for Evaluating Automatic Evaluation Metrics for Machine Translation, Proceedings of 20th International Conference on Computational Linguistic (COLING 2004), 2004.

[8] A.Amir, Z.Gotthilf, B.R.Shalom, Weighted LCS, *Journal of Discrete Algorithms*, **8**(2010), 273–281.

[9] S.V.Znamenskij, Modeling of the optimal sequence alignment problem, *Program systems: theory and applications*, **5**(2014), no. 4, 257–267.

[10] S.V.Znamenskij, A model and algorithm for sequence alignment, *Program systems: theory and applications*, **6**(2015), no. 1, 189–197 (in Russian).

Простые существенные улучшения алгоритма ROUGE-W

Сергей В. Знаменский

Алгоритм ROUGE-W для вычисления схожести текстов с 2004 года упоминается почти в 500 научных публикациях. Представлен оптимальный выбор весовой функции, от которой зависит эффективность алгоритма. Ранее использовались функции, далёкие от оптимальной. Приведён пример некорректного срабатывания алгоритма. Описаны несложные изменения в нём, гарантирующие ожидаемый результат.

Ключевые слова: длиннейшая общая подпоследовательность, ROUGE-W, выравнивание последовательностей, расстояние редактирования, схожесть строк, оптимизация, оценки сложности.