String Transformations Preserving Analogies

Yves Lepage
Graduate School of Information, Production and Systems
Waseda University
Kitakyushu, Japan
yves.lepage@waseda.jp

***Abstract*—This paper examines the following problem: which transformations on strings preserve analogy? For instance, consider the analogy putra : putera :: putri : puteri.[1] If we systematically reduplicate the characters in the strings (ppuuttrraa : ppuutteerraa :: ppuuttrrii : ppuutteerrii), or systematically insert a space between each character in the strings (p␣u␣t␣r␣a : p␣u␣t␣e␣r␣a :: p␣u␣t␣r␣i : p␣u␣t␣e␣r␣i), the analogies between the transformed strings still hold. The analogies considered are formal analogies of commutation between strings of characters, the definition of which makes use of LCS distance. Experiments on more than 16 million formal linguistic examples confirm several theoretical results, invalidate some hypotheses. and allow to test interesting conjectures.**

***Index Terms*—Strings, Analogy, Distance, String transformations**

I. Introduction

Formal analogical equations between strings of characters are puzzles of the type: aaabbbccc : aabbcc :: aabbcc : x (solution: abc) or kena : mengenakan :: keras : x (solution: mengeraskan).[2] Formal analogical equations differ from semantic analogies like: pria : wanita :: raja : x (solution: ratu)[3] or encyclopaedic analogies like: Indonesia : Jakarta :: Malaysia : x (solution: Kuala Lumpur). Data sets like Google set [12] or BATS v3.0 [5] have been created recently to test word embeddings [12], [13] on semantic or encyclopaedic analogies.

Algorithms to solve formal analogical equations have been proposed much earlier as in [9] or [8]. They are still under study [14]. To test them, large or very large test sets of formal analogies have also been released [14], [1], [6].

Recently, capitalising on the progress in deep learning, some attempts at solving formal analogical equations using neural networks have been made [7], [19]. Analogies between strings of characters are inherently variable in lengths but neural methods, especially fully connected networks, require fixed length input.

This poses the problem of casting a variable-length analogical puzzle into an equivalent puzzle where all strings, including the solution, have the same length. For instance, the puzzle aaabbbccc : aabbcc :: aabbcc : x contains strings of lengths 9, 6 and 6 respectively. It is obvious that a first transformation into a fixed-length puzzle aaabbbccc : aaabbbccc :: aaabbbccc : x' allows to build solution x' = aaabbbccc, of same length. A back-transformation yields the solution x = abc.

For the ultimate purpose of casting analogical equations into fixed-length puzzles so as to find the solutions of variable-length formal analogical equations, this paper examines which string transformations preserve analogies between strings, more precisely formal analogies of commutation (see Sect. IV-C) which are the most common in language data, and for which potential applications exist in transliteration, morphology and even machine translation.

[1]Indonesian: prince/son, princess/daughter in two possible spellings.
[2]Indonesian: close, bound : to bind :: stiff, firm : to stiffen.
[3]Indonesian: man : woman :: king : queen.

II. Prerequisites on Strings and Transformations

Let $\mathcal{A}$ be a set of characters, i.e., an alphabet. The set of all strings built on $\mathcal{A}$ is classically denoted as $\mathcal{A}^*$. It contains the empty string, noted with ε.

Table I gives the list of notations for strings used in this paper. In addition to standard operations like concatenation, mirroring (abc^R = cba) or repetition of strings (abc^3 = abcabcabc), we introduce two original operations:

- systematic reduplication of each character a certain number of times in a string (^{3}abc = aaabbbccc);
- systematic insertion of a string into another string after each character (abc $\downarrow$ xy = axybxycxy).

III. Prerequisites on LCS Edit Distance

A. Definition

A longest common subsequence (LCS) of two strings s_1 and s_2 is a longest possible string of characters which appear in the same order in the two strings. For instance, $\text{LCS}(\text{x}\mathbf{\text{a}}\text{y}\mathbf{\text{b}}\text{z}\mathbf{\text{c}}, \text{efg}\mathbf{\text{a}}\text{hi}\mathbf{\text{b}}\text{jklmop}\mathbf{\text{c}}\text{qr}) = \mathbf{\text{abc}}$. As illustrated here, an LCS can be made of discontiguous characters. There may exist several LCS for a given pair of strings. For instance aba, aca, acb, bab, bac, cab, etc., are all LCS of abcabc and cbacba; they all have the same length by definition.

The length of the LCS of two strings is a measure of their similarity. The smaller, the less similar the strings.

The LCS distance between two strings s_1 and s_2 is a distance defined from the LCS similarity. The larger, the less similar the strings. It can be directly defined as:

$$\text{d}(s_1, s_2) = |s_1| + |s_2| - 2 \times |\text{LCS}(s_1, s_2)| \qquad (1)$$

It is a true mathematical distance as it takes non-negative values and verifies the three following fundamental axioms for distances. For any strings s_1, s_2 and s_3 in $\mathcal{A}^*$,

TABLE I
NOTATIONS ON STRINGS USED IN THIS PAPER WITH EXAMPLES (GENERAL CASES FIRST, PARTICULAR CASES AFTER)

Definition	Notation	Examples		
Length of a string	$\|s\|$	$s = \text{mississippi}$	$\|\text{mississippi}\| = 11$	
		$s = \text{aaaa}$	$\|\text{aaaa}\| = 4$	
		$s = \varepsilon$	$\|\varepsilon\| = 0$	
Number of occurrences of character in a string	$\|s\|_c$	$s = \text{mississippi}$	$c = \text{s}$	$\|\text{mississippi}\|_\text{s} = 4$
		$s = \text{aaaa}$	$c = \text{a}$	$\|\text{aaaa}\|_\text{a} = 4$
		$s = \varepsilon$	$\forall c \in \mathcal{A}$	$\|\varepsilon\|_c = 0$
Set of characters in a string	$\overline{s}$	$s = \text{mississippi}$	$\overline{\text{mississippi}} = \{\text{i, m , p, s}\}$	
		$s = \text{aaaa}$	$\overline{\text{aaaa}} = \{\text{a}\}$	
		$s = \varepsilon$	$\overline{\varepsilon} = \emptyset$	
Concatenation of strings	$s_1.s_2$	$s_1 = \text{abc}$	$s_2 = \text{defgh}$	$s_1.s_2 = \text{abcdefgh}$
		$s_1 = \varepsilon$	$s_2 = \varepsilon$	$\varepsilon.\varepsilon = \varepsilon$
		$s_1 = \varepsilon$	$\forall s_2 \in \mathcal{A}^*$	$\varepsilon.s_2 = s_2$
		$\forall s_1 \in \mathcal{A}^*$	$s_2 = \varepsilon$	$s_1.\varepsilon = s_1$
Mirror of a string	s^R	$s = \text{abc}$	$\text{abc}^R = \text{cba}$	
		$s = \varepsilon$	$s^R = \varepsilon$	
		$s = \text{aabbaa}$	$s^R = \text{aabbaa} = s$	
Repetition of a string a certain number of times	s^λ	$\lambda = 3$	$s_1 = \text{abc}$	$\text{abc}^3 = \text{abcabcabc}$
		$\forall \lambda \in \mathbb{N}$	$s_1 = \varepsilon$	$\varepsilon^\lambda = \varepsilon$
		$\lambda = 0$	$\forall s \in \mathcal{A}^*$	$s^0 = \varepsilon$
		$\lambda = 1$	$\forall s \in \mathcal{A}^*$	$s^1 = s$
Systematic reduplication of each character in a string a certain number of times	${}^\lambda s$	$\lambda = 3$	$s = \text{abc}$	${}^3\text{abc} = \text{aaabbbccc}$
		$\forall \lambda \in \mathbb{N}$	$s = \varepsilon$	${}^\lambda\varepsilon = \varepsilon$
		$\lambda = 0$	$\forall s \in \mathcal{A}^*$	${}^0 s = \varepsilon$
		$\lambda = 1$	$\forall s \in \mathcal{A}^*$	${}^1 s = s$
Systematic insertion of a string into another string after each character	$s_1 \downarrow s_2$	$s_1 = \text{abc}$	$s_2 = \text{xy}$	$\text{abc} \downarrow \text{xy} = \text{axybxycxy}$
		$s_1 = \varepsilon$	$s_2 = \varepsilon$	$\varepsilon \downarrow \varepsilon = \varepsilon$
		$s_1 = \varepsilon$	$\forall s_2 \in \mathcal{A}^*$	$\varepsilon \downarrow s_2 = \varepsilon$
		$\forall s_1 \in \mathcal{A}^*$	$s_2 = \varepsilon$	$s_1 \downarrow \varepsilon = s_1$

- separation: $d(s_1, s_2) = 0 \Leftrightarrow s_1 = s_2$
- symmetry: $d(s_1, s_2) = d(s_2, s_1)$
- triangle inequality: $d(s_1, s_2) \leq d(s_1, s_3) + d(s_3, s_2)$

The LCS distance is also an edit distance, i.e., it is the smallest number of edit operations needed to transform one string into the other one. The LCS distance considers only two kinds of edit operations:

- insertion: ab $\rightarrow$ abc;
- deletion: abc $\rightarrow$ ab.

The LCS distance is different from the well-know Levenshtein distance [11] in which an additional kind of edit operation, substitution (abc $\rightarrow$ dbc), is used. When using the notion of cost of an edit operation, the LCS distance can be seen as a particular case where any substitution has a cost of two: the cost of one deletion plus the cost of one insertion.

The standard algorithm to compute edit distances is the dynamic programming algorithm given in [17] which uses an array representation of the problem. The time complexity is $O(|s_1| \times |s_2|)$ in the worst case, and in the best case. The complexity in the worst case seems difficult to beat and, according to [4], reducing it significantly would even be equivalent to prove that $P = NP$. There exist algorithms which improve speed in favorable cases for the LCS distance, indirectly [2] or directly [18], i.e., with or without computing the length of an LCS.

TABLE II
PROPERTIES OF LCS DISTANCE WITH RESPECT TO SOME OPERATIONS ON STRINGS

Operators		Property / Counter-example	Validity
Concatenation	$\forall(s_1, s_2, s_3) \in (\mathcal{A}^*)^3$	$\mathrm{d}(s_1.s_3, s_2.s_3) = \mathrm{d}(s_1, s_2)$ $\mathrm{d}(s_3.s_1, s_3.s_2) = \mathrm{d}(s_1, s_2)$	Always true Always true
Mirror	$\forall(s_1, s_2) \in (\mathcal{A}^*)^2$	$\mathrm{d}(s_1^R, s_2^R) = \mathrm{d}(s_1, s_2)$	Always true
Repetition	$\forall\lambda \in \mathbb{N}, \forall(s_1, s_2) \in (\mathcal{A}^*)^2$ $\lambda = 2, s_1 = ab, s_2 = ba$	$\mathrm{d}(s_1^\lambda, s_2^\lambda) = \lambda \times \mathrm{d}(s_1, s_2)$ $\mathrm{d}(abab, baba) = 2 \neq 2 \times 2$	Not true in general
Systematic Reduplication	$\forall(s_1, s_2) \in (\mathcal{A}^*)^2$	$\mathrm{d}({}^\lambda s_1, {}^\lambda s_2) = \lambda \times \mathrm{d}(s_1, s_2)$	Always true
Systematic Insertion	$\forall(s_1, s_2, s_3) \in (\mathcal{A}^*)^3$ $s_1 = ab, s_2 = ba, s_3 = ab$	$\mathrm{d}(s_1 \downarrow s_3, s_2 \downarrow s_3) = (\lvert s_3\rvert + 1) \times \mathrm{d}(s_1, s_2)$ $\mathrm{d}(aabbab, babaab) = 4 \neq 3 \times \mathrm{d}(ab, ba) = 3 \times 2$	Not true in general
	$\forall(s_1, s_2) \in (\mathcal{A}^*)^2, \forall c \in \mathcal{A} \setminus \overline{s_1.s_2}$ $s_1 = ab, s_2 = dbd, c = x$	$\mathrm{d}(s_1 \downarrow c, s_2 \downarrow c) = 2 \times \mathrm{d}(s_1, s_2)$ $\mathrm{d}(ab \downarrow c = axbx, dbd \downarrow c = dxbxdx) = 4 \neq \mathrm{d}(ab, dbd) = 3$	Not true in general

B. Transformations Preserving the LCS Distance

Table II lists a number of remarkable properties verified by the distance when applied to strings transformed by the operations introduced in Sect. III. It also includes some deceptive formulae and provides counter-examples for them.

To summarise the table, concatenation by the same string and mirroring preserve the LCS distance. Systematic reduplication behaves naturally in that it preserves the LCS distance up to a multiplicative factor, the reduplication factor. Succint proofs are given in Appendix A.

It is not a surprise that repetition does not preserve the LCS distance, even up to the multiplicative factor of the number of repetitions. Systematic insertion does not either behave well, even in the particular case where the inserted string reduces to one character which does not appear in any of the strings.

IV. PREREQUISITES ON ANALOGY

A. Definition

After having introduced the notions of strings, LCS distance, transformations on strings and their behaviour relative to LCS distance, we now turn to the core of our topic: analogies between strings. An analogy is defined in all generality as

> "a *conformity* of *ratios* between *objects* of the same type." [10]

A ratio is a quantitative measure for the comparison of two objects. Its symbol is a colon (:). A conformity is a resemblance relation [15], here between two ratios; it verifies reflexivity and symmetry, but not necessarily transitivity. Its symbol is a double colon (::). Ratio and conformity are the two *articulative* notions in analogy.

The above notations have a long tradition. With them, classically, an analogy is written $A : B :: C : D$ and reads "A is to B as C is to D."

There are also two *constitutive* notions in an analogy, namely contiguity and similarity.

Table III gives a set of axioms for analogies which sticks to the general definition of analogy given above. Axioms (I), (II) and (III) state that inversing conformity, ratios or objects preserves the analogy. Axioms (III) and (IV) concern the two main constitutive notions of analogy: contiguity and similarity. The last axiom, Axiom (V), has long been identified as being proper to analogy [3]; the tradition calls it exchange of the means, because B and C are called the *means;* A and D are called the *extremes.*

B. Types of Analogies on Strings

It is possible to distinguish four types of analogies on strings. By strings, for our applications in natural language processing, we mean words or sentences. The four types are:

- analogies of repetition. In language, they are classically illustrated by marked plurals in Indonesian or Malay: rumah : rumah-rumah :: kota : kota-kota;[4]
- analogies of reduplications. In morphology, reduplication of radical consonants are attested in Greek or Latin conjugation: do : dedi :: cado : cecidi.[5] In syntax, this is illustrated with comparative forms in Japanese: taka-i : taka-i hodo taka-kereba… :: naga-i : naga-i hodo naga-kereba…;[6]
- analogies of commutation. The simplest case, in reach of regular languages, is simple prefixing or suffixing: do : doing :: call : calling. Systematic parallel infixing as attested in Semitic languages makes the problem context-sensitive: kataba : kātib :: sakana : sākin;[7]
- analogies of mirroring. They are not attested in natural languages. They just consist in mirroring the strings: abc : cba :: defgh : hgfed.

More general analogies may merge several of the above types at the same time. For instance, xyabc : cba :: xydefgh : hgfed conflates an analogy of commutation with an analogy of mirroring. Some analogies belong to several types at the

[4]Indonesian or Malay: house : houses :: town : towns.
[5]Latin: I do : I did :: I fall : I fell.
[6]Japanese: high : the higher the… :: long : the longer the…
[7]Arabic: he wrote : writer :: he lived (in …) : inhabitant.

TABLE III
AXIOMS FOR ANALOGIES OF COMMUTATION BETWEEN STRINGS OF CHARACTERS

Number	Name of Axiom	Property
(O)	Reflexivity of *conformity*	$A : B :: A : B$ is always true.
(I)	Inversion of *conformity*	$A : B :: C : D \Rightarrow C : D :: A : B$
(II)	Inversion of *ratios*	$A : B :: C : D \Rightarrow B : A :: D : C$
(III)	Inversion of *objects* ($\Leftarrow$ contiguity)	$A : B :: C : D \Rightarrow A^{-1} : B^{-1} :: C^{-1} : D^{-1}$
(IV)	Distribution in *objects* ($\Leftarrow$ similarity)	$A : B :: C : D \Rightarrow$ any feature in A must be present in either B or C or both.
(V)	Exchange of the means	$A : B :: C : D \Rightarrow A : C :: B : D$

same time. For instance aa : aa :: aaa : aaa is an analogy of commutation as well as an analogy of mirroring and can be considered an analogy of reduplication.

We consider only analogies of commutation in this paper. This is because analogies of commutation are the most common type of analogy in language data, especially between short sentences. They exemplify patterns as in: My head hurts. : My head hurts badly :: His chest hurts. : His chest hurts badly. which are abundant in data sets like the BTEC [16] or the Tatoeba corpus[8].

C. Characterisation of Analogies of Commmutation

We use the characterisation of analogies between strings of characters from [10]. Basically, this characterisation states that the ratio between two strings A and B is composed of their LCS distance, $\mathrm{d}(A, B)$, and all the differences between the occurrences of all their characters, $|A|_c - |B|_c$, for all c in $\mathcal{A}$. The conformity of two ratios is established by the equality of each of these components. Writing the above with mathematical symbols gives Eq. (2) below.

$$\begin{aligned} &\forall (A, B, C, D) \in (\mathcal{A}^*)^4, \\ &\quad A :\ B ::\ C :\ D \ \Leftrightarrow \\ &\quad \begin{cases} \mathrm{d}(A, B) = \mathrm{d}(C, D) \\ \mathrm{d}(A, C) = \mathrm{d}(B, D) \\ |A|_c - |B|_c = |C|_c - |D|_c,\ \forall c \in \mathcal{A} \end{cases} \end{aligned} \tag{2}$$

It should be noted that the constraint on the number of characters trivially implies that the sum of the lengths of the extremes is equal to the sum of the lengths of the means.

$$|A| + |D| = |B| + |C| \tag{3}$$

This result can also be demonstrated from the two constraints on LCS distance.

V. TRANSFORMATIONS ON STRINGS PRESERVING ANALOGY

We are interested in determining which transformations t are such that:

$$A :\ B ::\ C :\ D \ \Rightarrow\ t(A) :\ t(B) ::\ t(C) :\ t(D) \tag{4}$$

We will examine three classes of cases. The first class is for the cases where we can provide proofs that the transformation preserves analogy. This is the object of Sect. *Theorems* below. The second class is for the cases where examples can be exhibited which show that the transformation does not preserve analogy in general. This is the object of Sect. *Experimental Refutations* below. The third class is for the cases where experimental evidence leads to think that a transformation should preserve analogy, but no formal proof has yet been provided. This is the object of Sect. *Conjectures* below.

A. Theorems

We start with theorems which can be easily established. They are given in Appendix B. They are summarised in the following way:

Prop. (4) holds for the following transformations:

- Max-length casting by prefixing or suffixing with a character outside of the set of characters appearing in the given strings;
- Mirroring;
- Systematic reduplication;

Max-length casting is not a transformation of the same kind as the other ones as the knowledge of the other strings in the analogy is necessary to parameterise it. However, we give it first as it is related to, but is not the same as, concatenation of strings by a common prefix (or suffix).

B. Experimental Refutations

We now turn to experimental refutations. We use the data released in [6][9]. It consists of 16,350,683 different analogies in ten different languages: Arabic, Finnish, Georgian, German, Hungarian, Maltese, Navajo, Russian, Spanish, Turkish.

The first hypothesis which we refute relates to repetition. It was shown in Sect. III-B that repetition does not preserve LCS distance. Hence, there is no surprise in experimentally invalidating the fact that an analogy between strings repeated the same number of times would follow from an analogy between these strings. However, on the very large data set

[8]https://tatoeba.org/

[9]http://lepage-lab.ips.waseda.ac.jp/, search for: 'SIGMORPHON data set'

TABLE IV

TRANSFORMATIONS AND THEIR SUCCESS RATE IN PRESERVING ANALOGIES ON A DATA SET OF 16 MILLION DIFFERENT ANALOGIES IN 10 DIFFERENT LANGUAGES. TOP ROWS: CONFIRMATION OF THEOREMS, MIDDLE ROWS: CONJECTURES, BOTTOM ROWS: REFUTATIONS OF HYPOTHESES

Transformation			Success
Name	Parameter	Analogy	rate (%)
Max-length casting	$c \in \mathcal{A} \setminus \overline{A.B.C.D}$, $m = \max(\lvert A\rvert, \lvert B\rvert, \lvert C\rvert, \lvert D\rvert)$	$c^{m-\lvert A\rvert}.A :: c^{m-\lvert B\rvert}.B :: c^{m-\lvert C\rvert}.C :: c^{m-\lvert D\rvert}.D$	100
Mirroring		$A.c^{m-\lvert A\rvert} :: B.c^{m-\lvert B\rvert} :: C.c^{m-\lvert C\rvert} :: D.c^{m-\lvert D\rvert}$	100
Systematic reduplication	$\lambda \in \{1, \ldots, 8\}$	$A^R :: B^R :: C^R :: D^R$	100
Systematic insertion	$c \in \mathcal{A} \setminus \overline{A.B.C.D}$	$A \downarrow c :: B \downarrow c :: C \downarrow c :: D \downarrow c$	100
Concaternation with opposite term's mirror		$A.D^R :: B.C^R :: C.B^R :: D.A^R$	100
Concatenation with own mirror		$A.A^R :: B.B^R :: C.C^R :: D.D^R$	98
Repetition	$\lambda = 2$	$A^\lambda :: B^\lambda :: C^\lambda :: D^\lambda$	90

used here, the success rate is surprisingly high, reaching 90 % for a repetition of 2.

The second hypothesis which we refute combines transformations. We consider the transformation which consists in concatenating a string with its mirror $s \rightarrow s.s^R$. On a small data set, we obtained a success rate of 100 % for the hypothesis that $A :: B :: C :: D$ would imply $A.A^R :: B.B^R :: C.C^R :: D.D^R$. Experiments on the very large data set used here invalidates it. However, the success rate is astonishingly high: 98 %. This deserves a future study of the cases where analogy is not preserved.

C. Conjectures

Against the negative result on systematic insertion not preserving LCS distance, we empirically tested whether systematic insertion of one character, outside of the set of characters found in the terms of the analogy, would preserve it. To much of our surprise, we observed a success rate of 100 % as reported on the first middle row in Table IV. The problem with the non-preservation of LCS distance is linked with the number of non-contiguous sequences in LCS. It may well be the case that analogy somehow enforces some balance there, so that Def. 2 is met. Still, this does not make a proof.

As an additional conjecture, we considered a transformation which requires the entire knowledge of the analogy, as was the case for max-length casting. We call opposite term of a string in an analogy the other string in the set of means or extremes, i.e., A is the opposite term of D and B is the opposite term of C, and reciprocally. The second middle row in Table IV stands for the test of this transformation, i.e: that $A :: B :: C :: D$ would imply $A.D^R :: B.C^R :: C.B^R :: D.A^R$. The success rate on the very large data set is 100 %.

It is in fact possible to give a theoretical proof of this second conjecture. In fact, $A.D^R :: B.C^R :: C.B^R :: D.A^R$ is an instance of $A' :: B' :: B'^R :: A'^R$ where all characters in A' and in B' are exactly the same with exactly the same number of occurrences. As $\mathrm{d}(A', B') = \mathrm{d}(B'^R, A'^R)$ and $\mathrm{d}(A', B'^R) = \mathrm{d}(B', A'^R)$ by properties of edit distances, the second conjecture can be proved a theorem.

VI. APPLICATION TO RESIZING ANALOGIES

The ultimate goal of this study is in fact to design transformations of analogies between variable-length strings into equivalent analogies between fixed-length strings for the purpose of their resolution. Let α denote the operation of solving an analogy, i.e., finding all strings D such that $A :: B :: C :: D$, given three strings A, B and C. It is a mapping from the set of triples of strings to the power set of the set of strings.

$$\begin{array}{rccl} \alpha : & (\mathcal{A}^*)^3 & \mapsto & 2^{\mathcal{A}^*} \\ & (A, B, C) & \rightarrow & \{D \in \mathcal{A}^* \ / \ A :: B :: C :: D\} \end{array} \tag{5}$$

This is justified by the two following facts. Firstly, formal analogical equations on strings may have several solutions. For instance, a : aa :: b : x has two solutions ab and ba when considered as an analogy of commutation[10] Secondly, formal analogical equations on strings may have no solution. For instance, ab : dk :: te : x has no solution.

We are interested in the following commutative diagram.

$$\begin{array}{ccc} (A, B, C) & \xrightarrow{\ t\ } & (t(A), t(B), t(C))) \\ \alpha \downarrow & & \downarrow \alpha \\ D & \xrightarrow{\qquad t \qquad} & t(D) \end{array} \tag{6}$$

where the transformation t makes all strings of the same length. The simple use of max-length casting leads to obvious possibilities, e.g., aaabbbccc : aabbcc :: aabbcc : $x \rightarrow$ aaabbbccc : xxxaabbcc :: xxxaabbcc : x'. Now, although still not clear at the moment, the conjecture on the concatenation with the opposite term's mirror may lead to fruitful considerations.

VII. CONCLUSION

We examined the problem of determining transformations on strings which preserve analogy. We restricted ourselves to formal analogies of commutation between strings of characters, for which we adopted a definition which mainly relies on LCS distance.

[10] And one, aa, if considered as an analogy of repetition.

Experiments on more than 16 million formal linguistic examples confirmed that max-length casting and systematic reduplication preserve analogy of commutation. The experiments also confirmed that repetition does not preserve analogy in general, but a high success rate was observed. Our framework allowed us to conjecture that, first, systematic insertion of a character not appearing in the strings, and, second, concatenation with the opposite term's mirror, also both preserve analogy. The proofs of on of these two conjectures remain an open problem.

ACKNOWLEDGEMENT

This research was supported in part by grant-in-aid n^o 18K11447 entitled "Self-explainable and fast-to-train example-based machine translation using neural networks.' from the Japanese Society for the Promotion of Science (JSPS).

APPENDIX

A. Proofs for Transformations Preserving LCS Edit Distance

1) Mirroring:

$$\forall (s_1, s_2) \in (\mathcal{A}^*)^2,\ \mathrm{d}(s_1^R, s_2^R) = \mathrm{d}(s_1, s_2) \tag{7}$$

For the LCS edit distance, the proof is obvious as the mirror of any longest common sub-sequence of two strings is the longest common sub-sequence of the two mirrored strings.

2) Systematic Reduplication:

$$\forall (s_1, s_2) \in (\mathcal{A}^*)^2,\ \mathrm{d}({}^{\lambda}s_1, {}^{\lambda}s_2) = \lambda \times \mathrm{d}(s_1, s_2) \tag{8}$$

The proof uses a lemma on the LCS length itself.

$$\forall (s_1, s_2) \in (\mathcal{A}^*)^2,\ |\mathrm{LCS}({}^{\lambda}s_1, {}^{\lambda}s_2)| = \lambda \times |\mathrm{LCS}(s_1, s_2)| \tag{9}$$

Prop.. (8) derives immediately from Lemma (9) and Def. (1) for LCS distance. The proof of Lemma (9) is as follows: a common substring of ${}^{\lambda}s_1$ and ${}^{\lambda}s_2$ is ${}^{\lambda}\mathrm{LCS}(s_1, s_2)$, of length $\lambda \times |\mathrm{LCS}(s_1, s_2)|$. Now, if an extra character can be added to it, this would imply that a character could have been added into $\mathrm{LCS}(s_1, s_2,)$, which contradicts the fact that $\mathrm{LCS}(s_1, s_2,)$ is a longest common sub-sequence. Hence, ${}^{\lambda}\mathrm{LCS}(s_1, s_2)$ is a longest common sub-sequence of s_1 and s_2.

B. Proofs for Transformations Preserving Analogy

1) Max-length casting: Given an analogy, it is always possible to prefix the strings with as many characters as needed to cast them into the maximum length of the strings. The character for prefixing is chosen outside of the set of characters appearing in the strings. The proof is easy. It firstly uses the fact that the LCS edit distance of two strings is the same as the LCS edit distance of these strings deprived of their common prefix (see Table II), and secondly Prop. (3).

2) Mirroring: Prop. (4) for mirroring derives from Prop. (7) and the fact that the number of occurrences of a character in a string is left unchanged through mirroring.

3) Systematic Reduplication: Prop. (4) for systematic reduplication derives from Prop. (8) and the fact that the number of occurrences of characters belonging to A, B, C and D is just multiplied by a factor of λ in ${}^{\lambda}A$, ${}^{\lambda}B$, ${}^{\lambda}C$ and ${}^{\lambda}D$. The constraint on number of occurrences of characters thus holds.

REFERENCES

[1] M. Abdou, A. Kulmizev, and V. Ravishankar. MGAD: Multilingual generation of analogy datasets. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[2] L. Allison and T. I. Dix. A bit string longest common subsequence algorithm. *Information Processing Letters*, 23:305–310, 1986.

[3] Aristotle. *Poetics*. Penguin, London, 1996. translated with an introduction and notes by M. Heath.

[4] A. Backurs and P. Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proceedings of the 47th Annual Symposium on the Theory of Computing (STOC'15)*, pages 51–58. ACM, 2015.

[5] A. Drozd, A. Gladkova, and S. Matsuoka. Word embeddings, analogies, and machine learning: Beyond king - man + woman = queen. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3519–3530. The COLING 2016 Organizing Committee, 2016.

[6] R. Fam and Y. Lepage. Tools for the production of analogical grids and a resource of n-gram analogical grids in 11 languages. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC 2018)*, pages 1060–1066, Miyazaki, Japan, May 2018. ELRA.

[7] V. Kaveeta and Y. Lepage. Solving analogical equations between strings of symbols using neural networks. In *Proceedings of the Computational Analogy Workshop at the 24th International Conference on Case-Based Reasoning (ICCBR-16)*, pages 67–76, Atlanta, Georgia, October 2016.

[8] P. Langlais, P. Zweigenbaum, and F. Yvon. Improvements in analogical learning: application to translating multi-terms of the medical domain. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2009)*, pages 487–495, Athens, Greece, March 2009. Association for Computational Linguistics.

[9] Y. Lepage. Solving analogies on words: an algorithm. In *COLING-ACL'98*, volume I, pages 728–735, Montréal, 1998.

[10] Y. Lepage. Proportional analogy in written language data. In N. Gala, R. Rapp, and G. Bel-Enguix, editors, *Language, Production, Cognition and the Lexicon*, Text, Speech and Language Technology 48, pages 1–23. Springer International Publishing Switzerland, 2014.

[11] V. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics-doklady*, 10(8):707–710, Feb. 1966.

[12] T. Mikolov, W.-T. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2013)*, pages 746–751, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[13] J. Pennington, R. Socher, and C. D. Manning. GloVe: Global vectors for word representation. In *Proceeedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[14] R. Rhouma and P. Langlais. Experiments in learning to solve formal analogical equations. In M. T. Cox, P. Funk, and S. Begum, editors, *Proceedings of the 26th International Conference on Case-Based Reasoning (ICCBR-18)*, pages 438–453, Stockholm, Sweden, August 2018. Springer.

[15] J. A. Schreider. *Equality, Resemblance, and Order*. Mir Publishers, 1975.

[16] T. Takezawa, E. Sumita, F. Sugaya, H. Yamamoto, and S. Yamamoto. Toward a broad coverage bilingual corpus for speech translation of travel conversation in the real world. In *Proceedings of LREC 2002*, pages 147–152, Las Palmas, May 2002.

[17] R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *Journal of the Association of Computing Machinery*, 21(1):168–173, Jan. 1974.

[18] S. Wu, U. Manber, G. Myers, and W. Miller. An O(NP) sequence comparison algorithm. *Information Processing Letters*, 35:317–323, April 1990.

[19] T. Zhao and Y. Lepage. Context encoder for analogies on strings. In *Proceedings of the 32th Pacific Asia Conference on Language, Information and Computation (PACLIC 32)*, pages ??–??, 2018.