TR-IT-0267

On Some Properties of Languages of Analogical Strings

Yves Lepage

July 1998

Abstract

We show how to define some formal languages using analogy between strings of characters. We prove that any language of analogical strings has the bounded growth property, a property relevant to the modelling of natural languages. In addition, $\{a^n/n \geq 1\}$, $\{a^n b^n/n \geq 1\}$, and $\{a^n b^n c^n/n \geq 1\}$ are shown to be languages of analogical strings, each capable of being analyzed with a similar asymptotic behavior. Moreover, the crucial formal language $\{a^n b^m c^n d^m/n \geq 1 \wedge m \geq 1\}$ is shown to be a simple language of analogical strings.

Keywords

Analogy, languages, bounded growth, formal languages hierarchy.

Contents

1 Introduction

In this report, we show how to define some formal languages through the operation of analogy. Analogies between character strings, noted $A : B = C : D$, put four character strings into "proportions". They render an account of, for instance, *look* : *looked* = *walk* : *walked* or *fable* : *fabulous* = *miracle* : *miraculous*, on the character string level. They are not intended to deal directly with, for instance, *bird* : *wings* = *fish* : *fins*, which supposes knowledge about the world. Analogies may be read as equalities, like in,

$$\text{Arabic: } \textit{aslama} : \textit{muslimun} = \textit{arsala} : \textit{mursilun}\ ^{1}$$

or as equations to be solved, as in,

$$\textit{aslama} : \textit{muslimun} = \textit{arsala} : \mathtt{x} \quad \Rightarrow \quad \mathtt{x} = \textit{mursilun}\ .$$

The founders of modern linguistics, from Baudouin de Courtenay [Stankiewicz 86], to [Saussure 16, Part 3, Chap. 4], and others, like [Kuryłowicz 49], have all reflected on analogy. The operation has been deeply felt to be crucial in terms of language. This is also true in morphology, where the synchronic effect (*actorem* : *actor* = *honorem* : *honor*) may be separated from the diachronic effect of phonetical change (*honosem* → *honorem*), and has also been hypothesized to be true in syntax (Hermann Paul, Bloomfield, Itkonen).

Nowadays, at a time when the use of language modelling has become a legitimate option in linguistics [Mel'čuk 97, 2–4], it is striking that only a small number of proposals have been made for the modelisation of analogy, the rare exceptions being [Itkonen 94] and [Hoffman 95] out of linguistics[2]. This may come from the fact that the dominant stream in linguistics for years, a generative one, as an exception in the history

[1] *arsala* (he sent) and *aslama* (he converted [to Islam]) are 3rd person singular past-tense verbs; *mursilun* (a sender) and *muslimun* (a convert, *i.e.* a muslim) are nouns.

[2] We dismiss [Gentner 83] and the like, as being studies on analogy. Meaning shifts are admissible in daily life, not in science. The definition of, and the difference between, analogy (*e.g.*, *"An electron is to the nucleus as a planet is to the sun"*) and metaphor (*e.g.*, *"an atom is like the solar system"*) have been known for almost 2,500 years, since Aristotle's Poetics (Part XXI). Gentner's work is about metaphor, not analogy. Fortunately, the small world of AI is not irreparably ignorant of the Classics, as [Steinhart 94] testifies.

of linguistics, has explicitly rebutted analogy as a possible object of research (see [Itkonen & Haukioja 97, 132 and 136], for quotations from Chomsky).

2 A speech in the defense of analogy

The explicit rejection of any recourse to analogy for grammaticality was a consequence of the following two hypotheses:

- the innateness hypothesis;
- the context-freeness hypothesis.

We will briefly discuss the refutation of these two hypotheses which led to the failure of the generativist program, also announced in an early paper [Gross 79].

2.1 The innateness hypothesis

Let us quote [Itkonen & Haukioja 97]:

> Chomsky (1957) had claimed that there was no discovery procedure for grammars, i.e. no method for deriving grammars from linguistic data; and because any such method has to be analogical (or inductive) in character, it followed (or seemed to follow) that there was no use for analogy (or induction) in linguistics.

This may be sketched by the following implication:

$$\text{learning data are too small} \Rightarrow \begin{cases} \text{no induction, no analogy} \\ \text{grammar is innate} \end{cases}$$

The fall of the innateness hypothesis The innateness hypothesis has been refuted in two ways. The first has been intellectual refutation [Itkonen 94] which has relied on a large number of recent results from psychology; these results have shown that isomorphism, iconicity, and thus analogy, actually play a role in the process of learning a language. The second one has been experimental refutation [Steels 97], which has shown the emergence of syntax in machines performing the game of language and equipped only with a categorization device.

The proposed answer: analogy The failure of the innateness hypothesis is an opportunity for a come-back of analogy, as analogy has a long history in linguistics. As a matter of fact, analogy may be called central in the Arabic grammatical tradition; it was also of importance for the founders of modern linguistics (in the discovery of, or

in opposition with sound changes) in the Kazan school (Baudouin de Courtenay, Gruszewski), as well as in the structuralist school (Saussure, Paul, Bloomfield). It does not seem far-strectched to say that transformations, as introduced by (Harris, Sager, Salkoff), or as used to define sub-languages by invariance (Kittredge), are close to the notion of analogy.

As an important remark, and as a transition, it must be underlined that none of the above mentioned linguists ever pretended that analogy is the one and only device which explains the whole of human language, in contrast with the alleged hegemony of context-free rules.

2.2 The context-freeness hypothesis

The family of context-free languages appears in the Chomsky classification which we recall in the following array. The proper inclusion between three important families of languages is recalled after the type of rules used to characterize languages of each family and the type of automata needed to recognize them. Also, typical languages for each family, which are not members of the previous family, are mentioned.

$\begin{cases} A \to Ba \\ A \to a \end{cases}$	$\begin{cases} A \to BC \\ A \to a \end{cases}$	$w \to w' \ / \ \mid w \mid \leq \mid w' \mid$
det./non-det.	non-det.	linear-bounded
finite state	push-down	

regular	$\subset$	context-free	$\subset$	context-sensitive
$\{a^n / n > 0\}$		$\{a^n b^n / n > 0\}$		$\{a^n b^n c^n / n > 0\}$
				$\{a^n b^m c^n d^m / n, m > 0\}$

The context-freeness hypothesis, which says that human languages belong to the context-free family, originates in the idea that the order of words *is* a syntactic structure. And because context-free languages have the property that any cut in the parse tree is a sentential expression, it was thought that formal languages of this type would be adequate to render an account of the constituency organization of natural languages.

However, constituency seems to be rather an exception among the large number of human languages. [Mel'čuk 88, 4–5] argues:

> To promote PS-representation in syntax, one has to be under the overall influence of English, with its rigid word order and almost total lack of syntactically driven morphology. However,

> English is very exotic in that it uses constituency almost as its only expressive device in syntax.

Hence the explicit accusation by [Itkonen 94]:

> Chomsky practiced some sort of 'universal grammar of English', taking the syntax of his native language to be an innate component of the human mind.

which may be summarized in the following way, where the truth of the first premise still remains to be proved, and the falsity of the second premise is obvious:

$$\left.\begin{array}{l}\text{English is context-free}\\ \text{English is universal}\end{array}\right\} \Rightarrow \text{ any human language is context-free}$$

The fall of the context-freeness hypothesis The context-freeness hypothesis was ruined by the discovery of a similar phenomenon, in morphology and in syntax, in two different languages.

The proof relies on the fact that a certain construction may be seen as the intersection of a regular language, $\{a^*b^*c^*d^*\}$, with the natural language $\mathcal{L}$ in question. Because the intersection of a context-free language with a regular language is known to be a context-free language, and because the said construction belongs to the family of context-sensitive languages, it must be concluded that the language in question is a member of the family of context-sensitive languages.

$$\mathcal{L} \cap \{a^*b^*c^*d^*\} = \{a^n b^m c^n d^m\} \text{ is a CSL} \Rightarrow \mathcal{L} \text{ is a CSL.}$$

The actual constructions were as follows.

Morphology of Bambara [Culy 85]

wulu	*nyinina*^{n}	*filèla*^{m}	*o wulu(nyinina)*^{n}*(filèla)*^{m}
dog	one who searches for	one who watches	whoever

"whoever is (one who watches)^m (searchers for)^n dog"

Syntax of the Zurich dialect of Swiss-German [Shieber 85]

(d'chind)^n	*(em Hans)*^{m}	*es huus haend wele*	*laa*^{n}	*hälfe*^{m}	*aastriche*
the children-ACC	Hans-DAT	the house-ACC have wanted	let	help	paint

"[...that we] have wanted to (let the children)^n (help Hans)^m paint the house"

The proposed answer: mild context-sensitivity As a consequence, the family of formal languages to formalize a natural language has to be larger than the family of context-free languages, but it does

not have not to cover all context-sensitive languages, as some context-sensitive languages are obviously not relevant for natural languages. Hence, the proposed term of *mild context-sensitivity*. [Joshi 85] proposed that the family of languages captured by tree-adjoining grammars would be the answer to the problem, since that family properly contains the family of context-free languages and is also strictly smaller than the family of context-sensitive languages.

However, this is a characterization by a recognition device, and some have proposed other tentative intrinsic characterizations of natural languages.

- A double characterization:
  - Semi-linearity (the Parikh set is a finite union of linear sets);
  - Parsability in polynomial time.
- A single characterization:
  - Bounded growth: for each sentence in a language, there is another sentence in the same language whose length differs from the length of the first sentence by at most a given constant.

We shall retain the second characterization for the sequel of this report.

2.3 Analogy and mild context-sensitivity

The goal of this report is therefore to constructively advocate for analogy by:

1. proposing an actual algorithm for analogy;
2. showing that this algorithm verifies some intuitive properties of analogy;
3. defining languages of analogical strings;
4. showing that languages of analogical strings may adequately describe natural languages by proving that:
   (a) languages of analogical strings have the bounded growth property;
   (b) the crucial language $\{a^n b^m c^n d^m / n > 0\}$ is a language of analogical strings.

All of this will demonstrate, we hope, the adequacy of languages of analogical strings for the description of natural languages, and hence, the usefulness of the proposed algorithm for natural language processing.

3 Analogy

3.1 Foundations of the algorithm

The seed for our algorithm is given by [Itkonen & Haukioja 97], who introduce a program in Prolog to obtain analogies on sentences with the help of structural information.

> Sentence D is formed by going through sentences B and C one element at a time and inspecting the relations of each element to the structure of sentence A (plus the part of sentence D that is ready).

Hence, sentence A is the axis against which sentences B and C are compared, and by opposition to which output sentence D is built.

reader : *unreadable* = $\overline{do}$*er* : x ⇒ x = *un*$\overline{do}$*able*

The method will therefore be: (a) look for those parts which are not common to A and B on one hand, and not common to A and C on the other and (b) put them together in the right order.

Looking for common subsequences of A and B (resp. A and C) solves problem (a) by complementation. The method of [Wagner & Fischer 74] finds the longest common subsequences by computing edit distance matrices, which yield the minimal number of edit operations (insertions, deletions, substitutions) necessary to transform one string into another.

For instance, the following matrices give the distance between *like* and *unlike* on one hand, and between *like* and *known* on the other hand, in their right bottom cells: dist(*like*, *unlike*) = 2 and dist(*like*, *known*) = 5

		u	*n*	*l*	*i*	*k*	*e*
l	1	2	2	3	4	5	
i	2	2	3	2	3	4	
k	3	3	3	3	2	3	
e	4	4	4	4	3	2	

		k	*n*	*o*	*w*	*n*
l	1	2	3	4	5	
i	2	2	3	4	5	
k	2	3	3	4	5	
e	3	3	4	4	5	

We call *similitude* between A and B the length of their longest common subsequence. It is also equal to the absolute length of A, minus the number of its characters deleted or replaced to produce B; this number we call pdist(A, B), because it is a pseudo-distance, which

can be computed exactly as the edit distances, except that insertions cost 0.

$$\mathrm{sim}(A, B) = \mid A \mid - \mathrm{pdist}(A, B)$$

For instance, $\mathrm{pdist}(\textit{unlike}, \textit{like}) = 2$, while $\mathrm{pdist}(\textit{like}, \textit{unlike}) = 0$.

	l	i	k	e
u	1	1	1	1
n	2	2	2	2
l	2	2	2	2
i	3	2	2	2
k	4	3	2	2
e	5	4	3	2

	u	n	l	i	k	e
l	1	1	0	0	0	0
i	2	2	1	0	0	0
k	3	3	2	1	0	0
e	4	4	3	2	1	0

Characters inserted into B or C may be left aside, precisely because they are those characters of B and C, absent from A, that we want to assemble into the solution, D.

As A is the axis in the resolution of analogy, graphically we make it the vertical axis around which the computation of pseudo-distances takes place. For instance, for $\textit{like} : \textit{unlike} = \textit{known} : \mathtt{x}$,

n	w	o	n	k		u	n	l	i	k	e
1	1	1	1	1	l	1	1	0	0	0	0
2	2	2	2	2	i	2	2	1	0	0	0
2	2	2	2	2	k	3	3	2	1	0	0
3	3	3	3	3	e	4	4	3	2	1	0

It is easy to verify that there is no solution to an analogy if some characters of A appear neither in B nor in C. The contrapositive says that, for an analogy to hold, any character of A has to appear in either B or C. Hence, the sum of the similitudes of A with B and C must be greater than or equal to its length: $\mathrm{sim}(A, B) + \mathrm{sim}(A, C) \geq \mid A \mid$, or, equivalently,

$$\mid A \mid \geq \mathrm{pdist}(A, B) + \mathrm{pdist}(A, C)$$

When the length of A is greater than the sum of the pseudo-distances, some subsequences of A are common to all strings in the same order. Such subsequences have to be copied into the solution D. We call

$\mathrm{com}(A, B, C, D)$ the sum of the lengths of such subsequences. The delicate point is that this sum depends precisely on the solution D being currently built by the algorithm.

To summarize, for analogy $A : B = C : D$ to hold, the following constraint must be verified:

$$| A | = \mathrm{pdist}(A, B) + \mathrm{pdist}(A, C) + \mathrm{com}(A, B, C, D)$$

3.2 The algorithm

Our method relies on the computation of two pseudo-distance matrices between the three first terms of the analogy. A result by [Ukkonen 85] says that it is sufficient to compute a diagonal band plus two extra bands on each of its sides in the edit distance matrix, in order to get the exact distance, if the value of the overall distance is known to be less than some given threshold. This result applies to pseudo-distances, and is used to reduce the computation of the two pseudo-distance matrices. The width of the extra bands is obtained by attempting to satisfy the coverage constraint with the value of the current pseudo-distance in the other matrix.

```
proc compute_matrices(A, B, C, pd_AB, pd_AC)
   compute pseudo-distances matrices with
   extra bands of pd_AB/2 and pd_AC/2
   if | A | ≥ pdist(A, B) + pdist(A, C)
      main component
   else
      compute_matrices(A, B, C,
          max(| A | − pdist(A, C), pd_AB + 1),
          max(| A | − pdist(A, B), pd_AC + 1))
   end if
end proc compute_matrices
```

Once enough cells in the matrices have been computed, the principle of the algorithm is to follow the paths along which the longest common subsequences are found, simultaneously in both matrices, copying characters into the solution accordingly. At each time point, the positions in both matrices must be on the same horizontal line, *i.e.* at the same position in A, in order to ensure a right order while building the solution, D.

The paths are determined by comparing the current cell in the matrix with its three previous ones (horizontal, vertical and diagonal), according to the technique in [Wagner & Fischer 74]. As a consequence, these paths are followed from the end of words down to their beginning. The nine possible combinations (three directions in two matrices) can be divided into two groups: either the directions are the same in both matrices, or they are different.

The following sketches the algorithm. $\mathrm{com}(A, B, C, D)$ has been initialized to: $\mid A \mid - (\mathrm{pdist}(A, B) + \mathrm{pdist}(A, C))$. i_A, i_B and i_C are the current positions in A, B and C. dir_{AB} (resp. dir_{AC}) is the direction of the path in matrix $A \times B$ (resp. $A \times C$) from the current position. "copy" means to copy a character from a word at the beginning of D and to move to the previous character in that word.

```
if constraint(i_A, i_B, i_C, com(A, B, C, D))
   case: dir_AB = dir_AC = diagonal
      if A[i_A] = B[i_B] = C[i_C]
         decrement com(A, B, C, D)
      end if
      copy B[i_B] + C[i_C] - A[i_A]^a
   case: dir_AB = dir_AC = horizontal
      copy char^b / min(pdist(A[1..i_A], B[1..i_B]),
                        pdist(A[1..i_A], C[1..i_C]))
   case: dir_AB = dir_AC = vertical
      move only in A (change horizontal line)
   case: dir_AB ≠ dir_AC
      if dir_AB = horizontal
         copy B[i_B]
      else if dir_AB = vertical
         move in A and C
      else same thing by exchanging B and C
      end if
end if
```

[a] In this case, we move in the three words at the same time. Also, the character arithmetics factors, in view of generalisations, different operations: if the three current characters in A, B and C are equal, copy this character; otherwise, copy that character from B or C that is different from the one in A; if all current characters are different, this is a failure.

[b] The word with less similitude with A is chosen, so as to make up for its delay.

An Example We will show how the analogy *like* : *unlike* = *known* : x is solved by the algorithm.

The algorithm first verifies that all letters of *like* are present either in *unlike* or *known*. Then, the minimum computation is done for the pseudo-distances matrices, *i.e.* only the minimal diagonal band is computed.

	e	*k*	*i*	*l*	*n*	*u*		*k*	*n*	*o*	*w*	*n*
.	.	.	.	0	1	1	*l*	1	1	.	.	.
.	.	.	0	1	2	.	*i*	.	2	2	.	.
.	.	0	1	2	.	.	*k*	.	.	3	3	.
0	1	2	.	.	.	.	*e*	.	.	.	4	4

As the coverage constraint is verified, the main component is called. It follows the paths noted by values in circles in the matrices.

	e	*k*	*i*	*l*	*n*	*u*		*k*	*n*	*o*	*w*	*n*
.	.	.	.	⓪	①	①	*l*	①	①	.	.	.
.	.	.	⓪	1	2	.	*i*	.	2	②	.	.
.	.	⓪	1	2	.	.	*k*	.	.	3	③	.
⓪	1	2	.	.	.	.	*e*	.	.	.	4	④

The succession of moves triggers the following copies into the solution:

dir_{AB}	dir_{AC}	copy
diagonal	diagonal	*n*
diagonal	diagonal	*w*
diagonal	diagonal	*o*
diagonal	diagonal	*n*
horizontal	horizontal	*k*
horizontal	diagonal	*n*
horizontal	diagonal	*u*

At each step, the coverage constraint is verified, and finally, the solution x = *unknown* is ouptut.

4 Some properties of analogy

In what follows, we take analogy as being defined mechanically by the algorithm just presented ([Lepage 98] or [Lepage & Iida 98]). This algorithm exhibits the following properties.

4.1 Exchange of the means and member inversion

Property 1 (Equivalent analogies) *Let the analogy* $A : B = C : D$ *hold. Then, the following analogies also hold:*

$A : C = B : D$	*by exchange of the means;*
$B : A = D : C$	*by member inversion;*
$D : B = C : A$	*by exchange of the extremes (or by exchange of the means, member inversion and symmetry of equality);*
$B : D = A : C$	*by member inversion and exchange of the means.*

The first and last analogies are the same.

Proof: The "exchange of the means" holds for the analogy we present, as, in our algorithm, B and C play an interchangeable role.

The analogy by member inversion may be derived using the exchange of the means if the property $A : B = C : D \Leftrightarrow C : D = A : B$ is verified. As this constraint may be easily added as a checking component in the algorithm, we take it for granted. Now, $A : B = C : D$ $\Leftrightarrow$ (exchange of the means) $A : C = B : D$ $\Leftrightarrow$ (symmetry of equality) $B : D = A : C$ $\Leftrightarrow$ (exchange of the means) $B : A = D : C$. **QED.**

These equivalent forms are common sense in terms analogy; Aristotle himself wrote the following in the Nicomachean Ethics (Book V):

> For proportion is equality of ratios, and involves four terms at least [...] As the term A, then, is to B, so will C be to D, and therefore, alternando, as A is to C, B will be to D. [Translation by W. D. Ross]

4.2 Constraints

There is no solution to an analogy $A : B = C : \mathtt{x}$ if some characters of A appear neither in B nor in C. The contrapositive says that, for an analogy to hold, any character of A has to appear in either B or C. If we note by $\overline{A}$ the set of characters contained in A, then, we have:

Property 2 (Character inclusion) *Let $\mathcal{V}$ be an alphabet,*

$$\forall (A,B,C,D) \in (\mathcal{V}^*)^4, \quad A:B = C:D \quad \Rightarrow \quad \overline{A} \subset \overline{B} \cup \overline{C}$$

The *similitude* between A and B is defined as the length of their longest common subsequence. It is also equal to the length of A, minus the number of its characters deleted or replaced to produce B. This last number we call $\mathrm{pdist}(A,B)$, because it is a pseudo-distance[3], which can be computed exactly as the edit distances ([Levenshtein 65] and [Wagner & Fischer 74]), except that insertions cost 0.

$$\mathrm{sim}(A,B) = \mid A \mid - \mathrm{pdist}(A,B)$$

The character inclusion property can be made more precise by saying that, the sum of the similitudes of A with B and C must be greater than or equal to its length: $\mathrm{sim}(A,B) + \mathrm{sim}(A,C) \quad \geq \quad \mid A \mid$, or, equivalently,

$$\mid A \mid \geq \mathrm{pdist}(A,B) + \mathrm{pdist}(A,C)$$

When the length of A is greater than the sum of the pseudo-distances, some subsequences of A are common to all strings, A, B, and C in the same order. As such subsequences are necessarily copied into the solution D, they also belong to D. We call $\mathrm{com}(A,B,C,D)$ the sum of the lengths of such subsequences. The difficult point is that this sum depends precisely on the solution D currently being built by the algorithm.

As a result, the following property is verified:

Property 3 (Analogy constraint) *Let $\mathcal{V}$ be an alphabet,* $\forall (A,B,C,D) \in (\mathcal{V}^*)^4$,

$$A:B = C:D \quad \Leftrightarrow \quad \mid A \mid = \mathrm{pdist}(A,B) + \mathrm{pdist}(A,C) + \mathrm{com}(A,B,C,D)$$

From the previous property, the following property is easily derived.

Property 4 (Equality of pseudo-distances) *Let $\mathcal{V}$ be an alphabet,*

$$\forall (A,B,C,D) \in (\mathcal{V}^*)^4, \quad A:B = C:D \quad \Rightarrow \quad \mathrm{pdist}(A,B) = \mathrm{pdist}(C,D)$$

[3]Firstly, $\mathrm{pdist}(A,B) = 0 \not\Rightarrow A = B$, as $\mathrm{pdist}(aaa, ababa) = 0$ illustrates. Secondly, $\mathrm{pdist}(A,B) \neq \mathrm{pdist}(B,A)$ in general, as $\mathrm{pdist}(aaa, ababa) = 0 \neq \mathrm{pdist}(ababa, aaa) = 2$ illustrates.

Proof: According to the Analogy constraint:

$$| A | = \text{pdist}(A, B) + \text{pdist}(A, C) + \text{com}(A, B, C, D)$$

By the definition of $\text{sim}(A, B)$,

$$| A | + \text{com}(A, B, C, D) = \text{sim}(A, B) + \text{sim}(A, C) \qquad (1)$$

Similarly, by the exchange of the means,

$$| C | + \text{com}(A, B, C, D) = \text{sim}(C, A) + \text{sim}(C, D)$$

Because $\text{sim}(A, B) = \text{sim}(B, A)$

$$| C | + \text{com}(A, B, C, D) = \text{sim}(A, C) + \text{sim}(C, D) \qquad (2)$$

By the subtraction of equalities (1) and (2):

$$| A | - \text{sim}(A, B) = | C | - \text{sim}(C, D)$$

By the definition of the pseudo-distances:

$$\text{pdist}(A, B) = \text{pdist}(C, D)$$

QED.

4.3 Concatenation of disjoint analogies

By disjoint analogies, we mean analogies such that the sets of characters involved in these analogies are disjoint. The solution of the concatenation of such analogies, is the concatenation of the solutions of each analogy.

Property 5 (Concatenation of disjoint analogies) *Let $\mathcal{V}$ be an alphabet,*
$\forall (A_1, B_1, C_1, D_1, A_2, B_2, C_2, D_2) \in (\mathcal{V}^*)^8$,

$$\left.\begin{array}{l} A_1 : B_1 = C_1 : D_1 \\ A_2 : B_2 = C_2 : D_2 \\ (\overline{A_1} \cup \overline{B_1} \cup \overline{C_1} \cup \overline{D_1}) \cap \\ \qquad (\overline{A_2} \cup \overline{B_2} \cup \overline{C_2} \cup \overline{D_2}) \;=\; \emptyset \end{array}\right\} \Rightarrow \quad A_1A_2 : B_1B_2 = C_1C_2 : D_1D_2$$

Proof: The algorithm relies on the computation of two matrices of pseudo-distances between A_1A_2 and B_1B_2 and between A_1A_2 and C_1C_2. Each of these matrices can be cut down into quadrants which may be designated by the indices (1,1), (1,2), (2,1), and (2,2). The quadrant (1,2), for instance, corresponds to the pseudo-distances between the substrings A_1 and B_2 in the context of A_1A_2 and B_1B_2.

If the disjoint hypothesis holds, it is easy to show that, necessarily:

$$\text{pdist}(A_1A_2, B_1B_2) = \text{pdist}(A_1, B_1) + \text{pdist}(A_2, B_2)$$

This result is also valid for any prefixes of A_2 and C_2.

In addition, because of the disjoint hypothesis, any $\text{com}(A_1a_2, B_1b_2, C_1c_2, D_1d_2)$ is necessarily equal to $\text{com}(A_1, B_1, C_1, D_1) + \text{com}(a_2, b_2, c_2, d_2)$, where a_2, b_2, c_2, and d_2 are prefixes of A_2, B_2, C_2, and D_2 respectively. As a result, the analogy constraint spells out the following:

$$\begin{aligned} |\, A_1a_2 \,| &= |\, A_1 \,| + |\, a_2 \,| \\ &= \text{pdist}(A_1a_2, B_1b_2) + \text{pdist}(A_1a_2, C_1c_2) + \text{com}(A_1a_2, B_1b_2, C_1c_2, \mathtt{x}') \\ &= \text{pdist}(A_1, B_1) + \text{pdist}(a_2, b_2) + \text{pdist}(A_1, C_1) + \text{pdist}(a_2, c_2) \\ &\quad + \text{com}(A_1, B_1, C_1, \mathtt{x}_1) + \text{com}(a_2, b_2, c_2, \mathtt{x}'_2) \end{aligned}$$

with $\mathtt{x}' = \mathtt{x}_1\mathtt{x}'_2$. Hence,

$$\begin{aligned} |\, A_1 \,| + |\, a_2 \,| = \quad & \text{pdist}(A_1, B_1) + \text{pdist}(A_1, C_1) + \text{com}(A_1, B_1, C_1, \mathtt{x}_1) \\ & + \text{pdist}(a_2, b_2) + \text{pdist}(a_2, c_2) + \text{com}(a_2, b_2, c_2, \mathtt{x}'_2) \end{aligned}$$

Because, by the hypothesis, $A_1 : B_1 = C_1 : D_1$ holds, the analogy constraint holds for $\mathtt{x}_1 = D_1$, and hence:

$$|\, a_2 \,| = \text{pdist}(a_2, b_2) + \text{pdist}(a_2, c_2) + \text{com}(a_2, b_2, c_2, \mathtt{x}'_2)$$

This means that the algorithm performs a task equivalent to solving the analogy $A_2 : B_2 = C_2 : \mathtt{x}_2$. By the hypothesis, the result is $\mathtt{x}_2 = D_2$.

To summarize: $\mathtt{x} = \mathtt{x}_1\mathtt{x}_2 = D_1D_2$.

QED.

This property meets the intuition that, analogies with nothing in common may be applied one after another without any problem.

5 Languages of analogical strings

5.1 Derivation

In order to show how some languages, *i.e.*, some sets of character strings, can be characterised by a device based on analogy, we first introduce the term "analogical derivation". We intentionally use this terminology to make a parallel with the vocabulary of formal grammars.

Definition 1 (Analogical derivation (modulo M)) *Let $\mathcal{V}$ be an alphabet. The analogical derivation modulo a set $\mathcal{M} \subset \mathcal{V}^* \times \mathcal{V}^*$, noted $\xrightarrow{\mathcal{M}}$, is defined in the following way:*

$$\forall (w, w') \in \mathcal{V}^* \times \mathcal{V}^*,\ w \xrightarrow{\mathcal{M}} w' \ \Leftrightarrow\ \exists (v, v') \in \mathcal{M} \ /\ \ w : w' = v : v'$$

This definition is tantamount to

- anchoring two terms of the analogy in a predefined set;
- reading the analogy as an equation to solve;
- defining the direction of the derivation, *i.e.*, that of the analogical equation, after the order given by the 2-tuples in $\mathcal{M}$.

5.2 Languages

Definition 2 (Language of analogical strings) *Let $\mathcal{V}$ be an alphabet. Let $\mathcal{A} \subset \mathcal{V}^*$ and $\mathcal{M} \subset \mathcal{V}^* \times \mathcal{V}^*$, then, the language of analogical strings $\Lambda(\mathcal{A}, \mathcal{M})$ is defined in the following way:*

$$\Lambda(\mathcal{A}, \mathcal{M}) = \mathcal{A} \cup \{\ w \in \mathcal{V}^* \ /\ \exists w' \in \mathcal{A} \ /\ w \xrightarrow{\mathcal{M}*} w'\ \}$$

with $\xrightarrow{\mathcal{M}*}$, the transitive closure of the analogical derivation $\xrightarrow{\mathcal{M}}$.

The interpretation is as follows: $\mathcal{A}$ is the set of attested strings, *i.e.*, the set of character strings against which the elements of the language are compared *in fine*, and $\mathcal{M}$ is the set of models, used to reduce[4] by analogy the elements of the language.

[4]The word *reduce* is taken to mean a reduction to a normal form, not in the sense that the strings become shorter.

In such a view, the "grammaticality", *i.e.*, the membership of a given string to the language, is tested against the set of attested strings of that language, after the reduction of that given string, by analogy, using the set of models. This approach, *i.e.*, by reduction to attested forms, using paradigms of declensions, conjugations, morphological derivations or syntactic transformations, seems close to the intuition about the way we, human beings, check the grammaticality of new sentences.

The previous reading is an analysis reading. Now, as member inversion is a property of analogy, we may choose to generate all of the members of a language of analogical strings, starting from the elements of $\mathcal{A}$, by applying all possible analogies with the elements of $\mathcal{M}$ as models. During this generation, the strings in the 2-tuples of $\mathcal{M}$ must be used in the reverse order they appear in $\mathcal{M}$. In the same way, to determine what the elementary language defined by the language of analogical strings is, we may proceed by induction, starting from the elements in $\mathcal{A}$.

Definition 3 (Simple language of analogical strings) *A language of analogical strings $\Lambda(\mathcal{A}, \mathcal{M})$ is said to be simple if and only if $\mathcal{A}$ is a singleton.*

In other words, for a simple language of analogical strings, there is only one string against which all other strings of the language are compared.

Definition 4 (Elementary language of analogical strings) *A language of analogical strings $\Lambda(\mathcal{A}, \mathcal{M})$ is said to be elementary if and only if $\mathcal{A}$ and $\mathcal{M}$ are singletons.*

In other words, for an elementary language of analogical strings, there is only one attested string, and there is only one model.

Examples of elementary languages of analogical strings are:

$$\begin{aligned}
\Lambda(\{a\}, \{(aa, a)\}) &= \{a^n \ / n \geq 1\} && \text{(regular language)} \\
\Lambda(\{ab\}, \{(aabb, ab)\}) &= \{a^n b^n \ / n \geq 1\} && \text{(context-free language)} \\
\Lambda(\{abc\}, \{(aabbcc, abc)\}) &= \{a^n b^n c^n \ / n \geq 1\} && \text{(context-sensitive language)}
\end{aligned}$$

We shall prove these equalities in the given sequel.

6 Some properties of languages of analogical strings

6.1 Bounded growth

[Marcus & al. 96] have been advocating that, the key point in "mild context-sensitivity" is in fact the property of bounded growth, rather than the two properties of semi-linearity and parsability in polynomial time.

Definition 5 (Bounded growth) *A language $\mathcal{L}$ has the bounded growth property if and only if*

$$\exists k \in \mathbb{N} \;/\; \forall w \in \mathcal{L},\ \exists w' \in \mathcal{L} \;/\; \| w \mid - \mid w' \| \leq k$$

We now prove that:

Theorem 1 *Any language of analogical strings verifies the bounded growth property.*

Proof: Let $\Lambda(\mathcal{A}, \mathcal{M})$ be a language of analogical strings. Let w be an element in $\Lambda(\mathcal{A}, \mathcal{M})$. By definition, unless $\Lambda(\mathcal{A}, \mathcal{M})$ is reduced to a singleton, in which case the bounded growth property trivially holds, there exists another element of $\Lambda(\mathcal{A}, \mathcal{M})$, and there exists $(v, v') \in \mathcal{M}$ such that $w : w' = v : v'$. The pseudo-distance equality property applies:

$$\text{pdist}(w, w') = \text{pdist}(v, v')$$

As the following property holds between the distance and pseudo-distance,

$$\text{dist}(w, w') \leq \text{pdist}(w, w') + \text{pdist}(w', w)$$

we get:

$$\begin{aligned} \forall w \in \Lambda(\mathcal{A}, \mathcal{M}), \exists w' \in \Lambda(\mathcal{A}, \mathcal{M}) \;/\; \| w \mid - \mid w' \| &\leq \text{dist}(w, w') \\ &\leq \text{pdist}(w, w') + \text{pdist}(w', w) \\ &\leq \text{pdist}(v, v') + \text{pdist}(v', v) \end{aligned}$$

We define k as being the maximum obtained over all elements of $\mathcal{M}$. With this,

$$\begin{aligned} \exists k = \max(\| v \mid - \mid v' \|, (v, v') \in \mathcal{M}) \quad / \\ \forall w \in \Lambda(\mathcal{A}, \mathcal{M}),\ \exists w' \in \Lambda(\mathcal{A}, \mathcal{M}) \quad / \quad \| w \mid - \mid w' \| \leq k \end{aligned}$$

QED.

6.2 The languages $\{a_1^n a_2^n \ldots a_m^n / n \geq 1\}$

6.2.1 The regular language $\{a^n / n \geq 1\}$

We now prove that the regular language $\{a^n\}$ is an elementary language of analogical strings.

Theorem 2 $\Lambda(\{a\}, \{(aa, a)\}) = \{a^n / n \geq 1\}$

Proof: In two steps, by showing the inclusion in both directions.

Let w be a string; recall that $\overline{w}$ is the set of different characters contained in w. If w belongs to $\Lambda(\{a\}, \{(aa, a)\})$, there must exist some $w_m, \ldots, w_1$ such that $w \xrightarrow{\mathcal{M}} w_m \xrightarrow{\mathcal{M}} \ldots \xrightarrow{\mathcal{M}} w_1 \xrightarrow{\mathcal{M}} a$. By the definition of a language of analogical strings, and because $\mathcal{M} = \{(aa, a)\}$ here, this is equivalent to:

$$w : w_m = aa : a \quad \wedge \quad w_m : w_{m-1} = aa : a \quad \wedge \quad \ldots \quad \wedge \quad w_1 : a = aa : a$$

By applying the Character inclusion property to these analogies, we get:

$$\overline{w} \subset \overline{w_m} \cup \overline{a} \subset \ldots \subset \overline{w_1} \cup \overline{a} \subset \overline{a} \cup \overline{a} = \{a\}$$

That is, $\overline{w} \subset \{a\}$, which implies that w is of the type a^n (note that there is no empty string here). Hence, $\Lambda(\{a\}, \{(aa, a)\}) \subset \{a^n / n \geq 1\}$.

Reciprocally, we prove by induction that any string of the form a^n can be obtained by analogy with an element of $\Lambda(\{a\}, \{(aa, a)\})$. This is true for a, as $\{a\} \subset \Lambda(\{a\}, \{(aa, a)\})$ by the definition of a language of analogical strings. It is also true for aa, as, trivially, $aa : a = aa : a$. Suppose now that a^{n-1} is a member of $\Lambda(\{a\}, \{(aa, a)\})$. We shall prove that the solution $\mathtt{x}$ of the analogy $a^n : \mathtt{x} = aa : a$ is a^{n-1}. By the Equality of pseudo-distances property:

$$\text{pdist}(a^n, \mathtt{x}) = \text{pdist}(aa, a) = 1$$

By the definition of pseudo-distances, $\mathtt{x}$ is therefore obtained from a^n either by deletion or by exchange of one character. Accordingly, we have either $\mathtt{x} = a^{n-1}$ or $\mathtt{x} = a^{i-1} b . a^{n-i}$ for some $b \in \mathcal{V}, b \neq a$ and some $i \in \mathbb{N}, i \leq n-1$.

$$\text{pdist}(a^n, aa) = n - 2 \neq \text{pdist}(a^{i-1} b . a^{n-i}, a) = n - 1$$

implies that the analogy $a^n : a^{i-1} b . a^{n-i} = aa : a$ is false, because the Equality of pseudo-distances property does not hold. Hence, $\mathtt{x} \neq$

$a^{i-1}b.a^{n-i}$. Consequently, $\mathbf{x} = a^{n-1}$ and from this, $a^n \in \Lambda(\{a\}, \{(aa, a)\})$, which concludes the induction. Hence,

$$\forall n \geq 1,\ a^n \in \Lambda(\{a\}, \{(aa, a)\})$$

or, equivalently, $\{a^n / n \geq 1\} \subset \Lambda(\{a\}, \{(aa, a)\})$.

The two previous inclusions prove that:

$$\Lambda(\{a\}, \{(aa, a)\}) = \{a^n / n \geq 1\}$$

QED.

6.2.2 The context-free language $\{a^n b^n / n \geq 1\}$

We now prove that the context-free language $\{a^n b^n\}$ is an elementary language of analogical strings.

Theorem 3 $\Lambda(\{ab\}, \{(aabb, ab)\}) = \{a^n b^n / n \geq 1\}$

Proof: By induction. By the definition of a language of analogical strings, $ab \in \Lambda(\{ab\}, \{(aabb, ab)\})$ is true. It is also true that $aabb \in \Lambda(\{ab\}, \{(aabb, ab)\})$ as the analogy $aabb : ab = aabb : ab$ trivially holds. Suppose that the following proposition is true for any $i > 1$ until a given n,

$$a^i b^i \in \Lambda(\{ab\}, \{(aabb, ab)\})$$

The proposition is proven true for n as follows: $a^n : a^{n-1} = aa : a$ and $b^n : b^{n-1} = bb : b$ are true analogies. By the Concatenation of disjoint analogies property, $a^n b^n : a^{n-1} b^{n-1} = aabb : ab$. As $a^{n-1} b^{n-1}$ is a member of $\Lambda(\{ab\}, \{(aabb, ab)\})$, the same also holds true for the case of $a^n b^n$. This concludes the proof by induction. **QED.**

6.2.3 The context-sensitive language $\{a^n b^n c^n / n \geq 1\}$

We now prove that the context-sensitive language $\{a^n b^n c^n\}$ is an elementary language of analogical strings.

Theorem 4 $\Lambda(\{abc\}, \{(aabbcc, abc)\}) = \{a^n b^n c^n / n \geq 1\}$

Proof: The proof is the same as for $\{a^n b^n / n \geq 1\}$, by decomposing

$$a^n b^n c^n : a^{n-1} b^{n-1} c^{n-1} = aabbcc : abc$$

into $a^n b^n : a^{n-1} b^{n-1} = aabb : ab$ and $c^n : c^{n-1} = cc : c$ which both hold.

QED.

More generally, the same proof may be used to show that:

$$\Lambda(\{a_1 a_2 \ldots a_m\}, \{(a_1^2 a_2^2 \ldots a_m^2, a_1 a_2 \ldots a_i)\}) = \{a_1^n a_2^n \ldots a_m^n / n \geq 1\}$$

6.2.4 Analysis of $\{a_1^n a_2^n \ldots a_m^n / n \geq 1\}$

Theorem 5 (Square complexity) *The recognition of* $a_1^n a_2^n \ldots a_m^n, n \geq 1$ *as an element of the language of analogical strings*

$$\Lambda(\{a_1 a_2 \ldots a_m\}, \{(a_1^2 a_2^2 \ldots a_m^2, a_1 a_2 \ldots a_m)\})$$

has an asymptotic behavior in $O(n^2)$.

Proof: We prove that the recognition can be performed in $m^2 \times n^2$ (cell) computations. The analogy resolution: $a_1^2 a_2^2 \ldots a_m^2 : a_1 a_2 \ldots a_m = a_1^k a_2^k \ldots a_m^k : \mathrm{x} \Rightarrow \mathrm{x} = a_1^{k-1} a_2^{k-1} \ldots a_m^{k-1}$ requires the computation of two pseudo-distance matrices. The pseudo-distance matrix between $a_1^2 a_2^2 \ldots a_m^2$ and $a_1^k a_2^k \ldots a_m^k$ contains $m \times 2 \times m \times k$ cells[5]. The pseudo-distance matrix between $a_1^2 a_2^2 \ldots a_m^2$ and $a_1 a_2 \ldots a_m$ can be computed once and stored. The actual generation of the solution of the analogy is done by following paths in both pseudo-distance matrices, and is hence performed in a time period linear in the length of the solution. Because this type of analogy is solved k times ($2 \leq k \leq n$), before obtaining $\mathrm{x} = a_1 a_2 \ldots a_m \in \{a_1 a_2 \ldots a_m\}$, the overall number of cells computed in all matrices, including the one between $a_1^2 a_2^2 \ldots a_m^2$ and $a_1 a_2 \ldots a_m$ is:

$$\sum_{k=1}^{n} m \times 2 \times m \times k = m^2 \times n \times (n-1)$$

[5] A result by [Ukkonen 85] shows that, in order to compute the exact distance between two strings, it is sufficient to compute a diagonal band plus two extra bands on each of its sides in the edit distance matrix, if the value of the overall distance is known to be less than some given threshold. The width of the extra bands grows linearly with the threshold. As mentioned in [Lepage 98], this result applies to pseudo-distances, and is used to reduce the computational load in pseudo-distance matrices, so that the actual overall computation, in the case of interest, is in fact strictly below the square.

The time of the actual generation of the solutions is also of the form:

$$\sum_{k=2}^{n} m \times (k-1) = m/2 \times (n-1) \times (n-2)$$

QED.

To summarize, as a remarkable result, the same asymptotic behavior is obtained (except for the multiplicative constant, which just depends on the number of different symbols):

- for any context-sensitive language of the general form $a_1^n a_2^n \ldots a_m^n$;
- for the context-free language $a^n b^n$;
- for the regular language a^n;

We consider it a good point, from the linguistic point of view, that the analysis complexity of all previous languages, when defined as languages of analogical strings, be the same. The reason is that there appears to be no difference in the complexity intuitively needed for the recognition of such languages, except, precisely, for the number of symbols involved. In this way, there is no unnatural distinction between $\{a^n b^n / n \geq 1\}$ and $\{a^n b^n c^n / n \geq 1\}$.

Of course, a square behavior seems "too much" for the analysis of the regular language $\{a^n / n \geq 1\}$. But the point is that the same complexity holds for all sorts of languages like $\{a^n / n \geq 1 \wedge p(n)\}$, where $p(n)$ may be any linear proposition on n like, to be odd, even, a multiple of a given integer, *etc.*

Reciprocally, a language like $\{a^n / n \geq 1 \wedge \exists m \in \mathbb{N} / n = 2^m\}$ does not seem to be easily represented, if it can be at all, by a language of analogical strings, and certainly not as an elementary language of analogical strings. This shows that some "unnatural" languages [Marcus & al. 96, 4] seem to be fortunately out of the reach of languages of analogical strings.

Somehow, analogy gets around the Chomsky classification. This is another, although converse, example of a phenomenon already mentioned in [Marcus & al. 96, 12], where two languages of the same complexity when generated by Chomsky grammars, are shown to be of a different complexity when generated by another device, namely contextual grammars.

6.3 The context-sensitive language $\{a^m b^n c^m d^n\}$

The language $\{a^m b^n c^m d^n\}$ is a simple language of analogical strings.

Theorem 6 $\Lambda(\{abcd\}, \{(abbcdd, abcd), (aabccd, abcd)\}) = \{a^m b^n c^m d^n / n \geq 1 \wedge m \geq 1\}$

The proof is easy by induction and use of the Concatenation of disjoint analogies.

This language is famous for being the corner-stone of two arguments in favor of the non-context-freeness of natural languages. The first of these two arguments is in morphology, more precisely in the morphology of Bambara [Culy 85], and the second argument is in the syntax of the Zurich dialect of Swiss German [Shieber 85]. Let us repeat what Shieber says at the end of his article:

> What has *not* been shown by this argument is equally important to keep in mind. By proving the non-context-freeness of the language of the Swiss-German competence grammar, we have still not demonstrated that natural languages are impossible, or even difficult, to parse. Both the Dutch and Swiss-German constructions are linear-parsable, and, were they not so in theory, performance constraints might well make them so.

We do not ensure the linearity of parsability (why should it be so?), but we have shown that, if competence relies on analogy, this language, considered as a language of analogical strings, can well be analysed by a competence grammar, without calling upon any performance constraint.

7 Conclusion

That, until now, analogy on strings of characters had been little if not at all studied, is understandable, if one considers the fact that the study of formal languages originates from the study of formal grammars, introduced by the generativist stream. As the generativist stream developed somehow in reaction to the structuralist stream, whose central tool was precisely analogy, and intended to replace it sociologically, it explicitly rejected analogy, and also induction, as a possible device for grammaticality, based on the common-sense experience that analogy in logic may lead to untruth, which has been known since the Sophists.

The fallacious pretext that blind application of analogy may lead to falsity (in logic) or agrammaticality (in syntax) is tantamount to saying that, any Chomsky grammar would be inadequate to describe a language, because some languages generated by some grammars include sentences not belonging to any natural language. Analogy is certainly not the "horizon indépassable" of linguistics, but it surely may be argued to be a component in language [Itkonen 94].

We have introduced a device, based on analogy, which permits some formal languages to be generated based on a set of models. It is important to note that analogical string grammars do not use any notion of non-terminals. The "grammaticality" of a given string, *i.e.*, its membership in a language, is tested against a set of attested strings of that language. This approach, by reduction to attested forms, seems close to the intuition about the way we, human beings, check the grammaticality of new sentences, and has already been advocated for in natural language processing [Salkoff 73].

We have proven a property of languages defined by this device, which, we think, may be a convincing argument in favor of the modelisation of a natural language by means of languages of analogical strings. This property, that of bounded growth, was proposed as an attempt to capture *mild context-sensitivity*, a notion introduced to cope with the apparent power of human languages, as exhibited by data from some specific natural languages.

The fact that the time complexity of the analysis of such languages as $\{a^n\}$, $\{a^n b^n\}$, $\{a^n b^n c^n\}$, which are all of a different power according to the Chomsky classification, be the same when regarded as languages of analogical strings, is, according to us, a strong argument in favor of analogy, as this seems closer to intuition. In other words, analogy

permits us to get around the Chomsky hierarchy, in the same way other devices, like, for instance, contextual grammars, do.

References

[Culy 85] Christopher Culy
The Complexity of the Vocabulary of Bambara
Linguistics and Philosophy, vol. 8, 1985, pp. 345-351.

[Dowty et al. 85] David R. Dowty, Lauri Kartunnen, Arnold M. Zwicky
Natural language processing
Cambridge University Press, Cambridge, 1985.

[Gentner 83] Dedre Gentner
Structure Mapping: A Theoretical Model for Analogy
Cognitive Science, 1983, vol. 7, no 2, pp. 155–170.

[Gross 79] Maurice Gross
On the Failure of Generative Grammar
Language, 55(4), pp. 859–85, 1979.

[Hoffman 95] Robert R. Hoffman
Monster Analogies
AI Magazine, Fall 1995, vol. 11, pp 11-35.

[Itkonen & Haukioja 97] Esa Itkonen & Jussi Haukioja
A rehabilitation of analogy in syntax (and elsewhere)
in András Kertész (ed.) *Metalinguistik im Wandel: die kognitive Wende in Wissenschaftstheorie und Linguistik* Frankfurt a/M, Peter Lang, 1997, pp. 131-177.

[Itkonen 94] Esa Itkonen
Iconicity, analogy, and universal grammar
Journal of Pragmatics, 1994, vol. 22, pp. 37-53.

[Joshi 85] Aravind K. Joshi
Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural description?
in [Dowty et al. 85], pp 206-250.

[Kudlek & al. 96] Manfred Kudlek, Solomon Marcus, Alexandru Mateescu
Contextual Grammars with Distributed Catenation and Shuffle

Turku center for Computer Science, TUCS technical Report No 44, March 1997.

[Kuryłowicz 49] Jerzy Kuryłowicz
La nature des procès dits « analogiques »
Acta Linguistica, 5, 1949, pp. 15-37.

[Lepage & Iida 98] Yves Lepage & 飯田仁
言語に依存しない早期終了型類推解決手法
言語処理学会第 4 回年次大会, 九州大学, 1998 年 3 月, pp. 266-269.

[Lepage 98] Yves Lepage
Solving Analogies between Words: an Algorithm
Proceedings of COLING-98, Montréal, August 1998, pp. ??–??.

[Levenshtein 65] V.I. Levenshtein
Binary codes capable of correcting deletions, insertions and reversals
Dokl. Akad. Nauk SSSR, vol. 163, No. 4, August 1965, pp. 845-848.
English translation in *Soviet Physics-doklady* vol. 10, No. 8, February 1966, pp. 707-710.

[Marcus & al. 96] Solomon Marcus, Carlos Martin-Vide, Gheorghe Păun
Contextual Grammars versus natural Languages
Turku center for Computer Science, TUCS technical Report No 44, Sept. 1996.

[Mel'čuk 97] Igor A. Mel'čuk
Vers une linguistique Sens-Texte. Leçon inaugurale.
Collège de France, Chaire internationale, 10 janvier 1997.

[Mel'čuk 88] Igor A. Mel'čuk
Dependency Syntax: Theory and Practice
State University of New York Press, 1988.

[Ukkonen 85] Esko Ukkonen
Algorithms for Approximate String Matching
Information and Control, **64**, 1985, pp. 100-118.

[Salkoff 73] Morris Salkoff
Une grammaire en chaîne du français
Dunod, 1973.

[Saussure 16] Ferdinand de Saussure
Cours de linguistique générale
publié par Charles Bally et Albert Sechehaye, Payot, Lausanne et Paris, 1916.

[Shieber 85] Stuart M. Shieber
Evidence against the Context-Freeness of Natural Language
Linguistics and Philosophy, vol. 8, 1985, pp. 333-343.

[Stankiewicz 86] Edward Stankiewicz
Baudouin de Courtenay i podstawy współczesnego językonawstwa
Ossolineum, Wrocław, 1986.

[Steels 97] Luc Steels
Origin of Syntax in Visually Grounded Robotic Agents
Proceedings of IJCAI-97, vol. 2, pp. 1632–1641, Nagoya, August 1997.

[Steinhart 94] Eric Steinhart
Analogical Truth Conditions for Metaphors
Metaphor and Symbolic Activity, 1994, 9(3), pp 161-178.

[Wagner & Fischer 74] Robert A. Wagner and Michael J. Fischer
The String-to-String Correction Problem
Journal for the Association of Computing Machinery, Vol. 21, No. 1, January 1974, pp. 168-173.

Index