Solving analogical equations on words

François Yvon
GET/ENST et LTCI, CNRS UMR 5141
46 rue Barrault
F-75013 Paris
yvon@enst.fr

Nicolas Stroppa
GET/ENST et LTCI, CNRS UMR 5141
46 rue Barrault
F-75013 Paris
stroppa@enst.fr

Arnaud Delhay
INRIA/CORDIAL
6, rue de Kerampont - BP 447
F-22305 Lannion Cedex
arnaud.delhay@univ-rennes1.fr

Laurent Miclet
INRIA/CORDIAL
6, rue de Kerampont - BP 447
F-22305 Lannion Cedex
laurent.miclet@enssat.fr

Résolution d'équations analogiques sur les mots

Résumé

Le raisonnement et l'apprentissage par analogies ont, de longue date, été proposés comme étant des modèles cognitivement plausibles de certaines capacités du cerveau humain. Ces modèles ont de plus donnés lieu à de multiples implantations de systèmes symboliques pour des tâches de résolution de problèmes. Compte tenu de la forte structuration interne des données linguistiques, les applications en traitement automatique des langues sont un champ d'application presque naturel de ces modèles. Pour mettre en œuvre de telles applications, il est toutefois nécessaire de disposer de dispositifs d'apprentissage capables de traiter de larges bases d'exemples, chaque exemple étant représenté comme une séquence, puisque telle est la forme que prennent le plus souvent les données linguistiques.

Dans cet article, nous présentons un cadre algébrique à l'aide duquel nous définissons la notion d'analogie entre quatre mots sur un alphabet fini. Quand un des mots est inconnu, une telle relation se transforme en une équation analogique. Nous présentons plusieurs algorithmes exacts et heuristiques pour calculer efficacement l'ensemble des solutions de telles équations. Nous montrons en particulier que notre définition d'une analogie entre mots conduit à des algorithmes de résolution fondés sur les transducteurs à états finis, sur la base desquels des procédures efficaces d'apprentissage par analogie peuvent être développées.

Solving Analogical Equations on Words

Abstract

Analogical reasoning and learning have long been presented as plausible models for some of the human mind abilities, with the additional bonus of yielding effective implementation of problem-solving devices operating on symbolic representations. Given the strong structural organization of linguistic data, Natural Language Processing applications are a natural test-bed for such learning device. Such applications however require learning devices to manipulate large databases of sequential data, which is the natural form under which linguistic data occurs.

In this paper, we present an algebraic framework for the definition of an analogical relationship between four words on a finite alphabet. When one word is unknown, this relation turns into to what is called an analogical equation. We present several exact and heuristic algorithms for efficiently computing the set of solutions to such equations. We show that our definition of an analogy between words yields finite-state analogy-solving machines, on the basis of which effective analogical learning procedures can be implemented.

1 Introduction

1.1 Motivations

1.1.1 Analogical reasoning and learning

Analogical Reasoning (AR) has a long standing tradition in Artificial Intelligence, justified and supported by a large body of empirical psychochological studies concerning the analogizing capacities of the human mind (see eg. [9]). The development of case-based reasoning systems [13] has somehow revived this tradition. In an nutshell, AR addresses the problem of solving a new problem based on some previous experiments assumed to be stored in some long term memory. The new and the past problems can address situations from apparently very remote domains, as for instance in [7], who examine the analogy betwen heat-flow and water-flow. Analogical reasoning follows the following main steps:

- *matching* of the new situation with the known past situations
- *selection* and *retrieval* of relevant past solutions
- *transfer* of the past solution to the new problem, based on an induced *mapping* between sub-parts of the old and new problems.

During each of these steps, existing domain knowledge can be utilized to enrich the inferential process.

As compared to similarity-based reasoning, analogical reasoning is characterized by the emphasis on *structural relationships* during the matching and transfer steps [6, 12]. Good matches should display a similarity in the relationships between the objects involved in the problem, rather than just a surface similarity. Studying the problem of solving analogical equations on words, [11] also puts the emphasis on the *dynamics of analogical processes*: matching and transfer should be based on a perception which emerges from the analysis of the problem.

Put in formal terms, the resolution of problem P requires to identify a *structural similarity* between P and a past problem P', whose solution $s(P')$ serves to infer a solution $s(P)$, on the ground that "P is to P' as $s(P)$ is to $s(P')$".

A simplistic implementation of this general framework is the categorization task, which consists in predicting a categorial output for some input data, based solely on the knowledge of past instances of similar (input, output) pairs. In the machine learning literature, this situation is formalized as follows: let $\mathcal{S}$ be a set of traning examples $\mathcal{S} = \{(x_i, f(x_i))_{i=1,m}\}$, where x_i is the representation of an example (taking the form of a vector of categorial or numerical features) and $f(x_i)$ the associated label, drawn from a closed list. Given the description t of a new pattern, the supervised classification task is to predict the label $f(t)$, based only on the knowledge of $\mathcal{S}$ (see eg. [24] for an introduction to the variety of existing answers to this problem).

This simple setting allows to illustrate the difference between similarity-based learning and analogical learning. A similarity-based approach, such as the k nearest-neighbour (k-nn) algorithm, finds, during the matching step, the k descriptions $x_1^* \dots x_k^*$ which minimize some predefined measure of distance to t; the transfer step then computes $f(t)$ based on the values $\{f(x_i^*), i = 1 \dots k\}$. In comparison, a possible instanciation of analogical learning would proceed along the following lines:

- search $\mathcal{S}$ for triples (x_i^*, y_i^*, z_i^*) such that $x_i^* : y_i^* :: z_i^* : t$. Searching analog 4-uplets offers a means to implicitely capture the structural similarity between t and known objects.
- predict $f(t)$ based on the solutions to $f(x_i^*) : f(y_i^*) :: f(z_i^*) :?$

In this approach, both the matching and the transfer steps try to take into account the internal structure of $\mathcal{S}$, so as to approximate the notion of structural similarity between t and known objects. In this paper, we are primarily concerned with the implementation of such analogical learners, which chiefly require efficient procedures for computing analogical relationships.

A general requirement for learning devices is their ability to cope with large databases of instances: practical implementations of the k-nn algorithm speed-up the matching step using efficient data structures for representing the memory [24]; the computation of distances can also be optimized [22]. The same efficiency is expected from analogical learners; this is especially crucial in our application domain, natural language processing, where training databases commonly contain hundreds of thousands instances.

1.1.2 Analogical learning for natural language processing

The supervised classification model sketched above has been successfully applied to a large variety of NLP tasks such as morphological analysis, part-of-speech tagging, surface syntactic analysis, word sense disambiguation... Various studies (eg. [26, 17, 14]) have made a strong case for analogical analyses of linguistic data. Indeed, the kinds of systematic alternation patterns which are observed in morphology, and to some extend, in syntax, indicates that many NLP tasks could profitably be approached with analogical reasonning or learning tools.

There is, in fact, another reason for considering analogical learners. In the NLP context, the standard classification model sometimes requires a significant preprocessing of the original data, aimed at producing the feature-based representation typically required by most categorization devices. However, linguistic data is sequential in nature, and does not always lend itself to this kinds of preprocessing: this is notably the case for complex data types such as sequences, trees, recursive feature structures, etc., whose representation as a set of categorial features is not always direct and may result in some loss of information.

Various attemps have thus been made to devise analogical learners based on simple and intuitive definitions of analogical relationships for sequential data. For instance, in [32], the input data consists of lexemes, described by their orthographic representation and their part-of-speech, the output to be produced is a linear string of phonemes. In this case, the inferential process translates graphemic strings into phonemics strings, the final application being text-to-speech synthesis. In [16], analogy is used to produce translations between sentences in different languages. Other attempts at analogical learning from word strings or tree-based representations include [8, 31, 15, 1, 10].

These attempts have shown that, as far as their generalization performance is concerned, analogical learners were on a par with other state-of-the-art techniques. Various limitations of the analogical learning paradigm have however been identified:

- too narrow a definition of analogical relationship often caused the matching and/or the transfer step to fail, causing a general failure of the inference mecanism;

- the inefficiency of the matching procedure seriously limitates the applicability of analogical learning from large databases.

The main motivation of this paper is to address such shortcomings, by providing a sound, gradual, and computationnaly tractable definition of an analogy between sequential data. This definition should additionnally be well adapted to linguistic data and that should yield efficient resolution procedures.

Our motivations thus sharply contrast with other attempts to model analogical relationships in the word domain, such as [23, 3], which are more concerned with defining a general model of analogical reasoning, capable of handling abstract and complex mappings. In these studies, the word domain is primarily used for illustration purposes, and not, as in our approach, as the main application domain.

1.2 Organization of the paper

The notion of analogy has a long history in philosophy and linguistics, and a formal definition should match the classical meaning and properties of this concept. For instance, any definition must assume that the "is to" relation is symmetrical. Section 2 presents some generalities on the notion of analogy, especially when applied to words. We will restrict our analysis to purely syntactic criteria, excluding semantic relations such as "`cow` *is to* `calf` *as* `mare` *is to* `foal`".

In Section 3, an initial formal definition of an analogy between sequences over of finite alphabet is given. This definition is based on rational operators, and allows an algebraic characterization of the set of solutions to any analogical equation, with the additional bonus of yielding an effective implementation under the form of finite-state transducers. A criterion measuring the quality of an analogical relation is also proposed, which enables us to sort the set of solutions to an analogical equation.

We then consider a more general situation (Section 4), where analogical relationships on words derive from arbitrary analogies on individual symbols, such as "`a` *is to* `A` *as* `b` *is to* `B`". We show that this extension of the model also leads to a finite-state automata based formulation. We finally discuss various contexts in which 'complex' analogies on individual symbols can appear.

In Section 5, we explore possible ways to rank competing analogies, and develop a heuristic resolution algorithm, which is a particular case of this general framework, aimed to reduce its algorithmic complexity. The basic idea is to reduce the search space based on an edit distance computation. We study the complexities of the general vs. the heuristic approach and give a characterization of the obtained solutions.

We contrast our approach with other attempts at the same problem in Section 6. Section 7 finally draws preliminary conclusions, lists open questions and discusses possible extensions of this work.

2 Analogy

2.1 A few examples

Let X be some set of objects and A, B, C and D four elements in X. An *analogy* on X is a triple of relations, the first one (an "is to" relation) between A and B, the second one (also an "is to" relation) between C and D, and the third one (of a different nature, called "as") between the manner that A is related to B and C is related to D. This analogy is denoted: $A : B :: C : D$, which reads "A is to B as C is to D". In the following, we give possible instanciations of this notion for simple object types.

Analogies in $\mathbb{R}^n$

Let X be the vector space $\mathbb{R}^n$ with the origin O. Then, if $\overrightarrow{A}$, $\overrightarrow{B}$, $\overrightarrow{C}$ and $\overrightarrow{D}$ are four elements in $\mathbb{R}^n$, an interpretation of an analogy $\overrightarrow{A} : \overrightarrow{B} :: \overrightarrow{C} : \overrightarrow{D}$ could be that A, B, C, D correspond to the summits of a parallelogram, A and D corresponding to opposite summits. In the form of an analogical equation:

$$\overrightarrow{A} - \overrightarrow{B} = \overrightarrow{C} - \overrightarrow{D}$$

which can be also written:

$$\overrightarrow{AB} = \overrightarrow{CD}$$

Analogies on finite sets

Let X be a family of finite sets and A, B, C and D four sets in X. An analogical relationship $A : B :: C : D$ can be defined as [18]:

$$A \cup D = B \cup C$$

which is equivalent to:

$$D = ((B \cup C) \backslash A) \cup (B \cap C)$$

provided that $A \subset B \cup C$ and $B \cap C \subset A$.

Analogies in linguistics

Even if they lack the kind of formal definition we are aiming at, linguistic studies provide us a rich source of examples of analogical relationships. For instance, analogy appears in the writings of F. de Saussure [4] as a principle of word form change. Through time, irregular word forms tend to be re-analysed based on analogies with more regular ones. An example is the evolution of the latin word pair *honōs* : *honorem*, which evolved into *honor* : *honorem*, a much more regular alternation pattern, as evidenced by eg. *orator* : *oratorem*. This example suggests that a valid analogical relationship is established as: *honor* : *honorem* :: *orator* : *oratorem*, where two different roots forms (honor and orator) alternate with two different suffixes (void and em). Examples of such alternations[1] abound in the morphological systems of indo-european languages,

[1] Simple prefix-suffix alternations may also occur by accident, as evidenced by the 4-uple *imp* : *imply* :: *great* : *greatly*. Such accidental analogies can be ruled out on the basis of additional syntactico-semantic properties (in this case, it suffices to observe that the corresponding part-of-speech labels (Noun, Verb, Adjective, Adverb) do not form a valid analogical proportion).

and can also be observed in syntactical data. The formalization of this kinds of relationships will thus be our primary focus.

Conversely, we will not consider here analogical relationships on semantic representations, such as: $cow : calf :: mare : foal$, which are independant of the orthographical representation. We believe that these can readily be modeled with analogies on sets, provided an appropriate feature-based semantic representation [29, 21].

2.2 Properties of analogical relations

There is no general definition of an analogical relation, the "is to" and the "as" relationships depending on the nature of X. However, according to the usual meaning of the word "analogy" in philosophy and linguistics, two basic axioms are generally required ([19]):

Symmetry of the "as" relation: $C : D :: A : B$
Exchange of the means: $A : C :: B : D$

As a consequence of these two axioms, five other formulations can be proven equivalent to $A : B :: C : D$:

Inversion of ratios: $B : A :: D : C$
Exchange of the extremes: $D : B :: C : A$
Symmetry of reading: $D : C :: B : A$
$B : D :: A : C$
$C : A :: D : B$

Another possible axiom (called *determinism*) requires that one of the two following trivial equations has a unique solution (the other being a consequence):

$$\begin{array}{lcl} A : A :: B : X & \Rightarrow & X = B \\ A : B :: A : X & \Rightarrow & X = B \end{array}$$

All the examples above conform to these three axioms. Our first goal in this article is thus to formulate a precise definition of this notion for the word domain. This is the topic we address in the next section.

3 Analogy on words

3.1 Notations

A word is a finite sequence of symbols from a finite alphabet Σ. ϵ is the empty word and $\Sigma^\star$ is the set of all words. We note $|x|$ the length of x; by definition we have $|\epsilon| = 0$. For $0 < i \leq |x|$, the symbol at position i in x is denoted $x(i)$. For x, y in $\Sigma^\star$, xy is the concatenation of x and y.

An (ordered) index set is a finite set of positive integers $i_1 \dots i_n$ such that $i_1 < i_2 \dots < i_n$. The set of all ordered index sets is denoted $\mathcal{I}$. If $I = \{i_1 \dots i_n\}$ and $J = \{j_1 \dots j_m\}$ are two disjoint index sets, $I + J$ denotes the ordered set containing the integers from $I \cup J$. Conversely, if $J \subset I$, $I \setminus J$ denotes the complementary of J in I. Sets of ordered consecutive integers from i to j will be denoted: $[\![i, j]\!]$. If u is a word, I_u denotes the index set: $[\![1, |u|]\!]$.

If $w = uzv$, we call u a *prefix* of w, v a *suffix* of w, and z a *factor* of w. A (scattered) *subword* of $w = w(1) \dots w(n)$ is a word v iif there exists an index set $S = \{i_1 \dots i_k\}$, st. $i_k < n$ and $w(i_1) \dots w(i_k) = v$. Conversely, an index set I included in $[\![1, n]\!]$ induces a subword of w, denoted $w(I)$. The subword relationship between v and w will be denoted $v \sqsubset w$. Prefixes, suffixes and factors are special types of subwords, induced by sets of *consecutive* indices.

A *factorization* of a word x is a sequence $x_1 x_2 \dots x_m$ such that $\forall i \in [\![1, m]\!]$, $x_i \in \Sigma^\star$ and $x_1 x_2 \dots x_m = x$.

A finite automaton A is a 5-tuple $(\Sigma, Q, q^0, F, \delta)$, where Σ is a finite alphabet, Q a finite set of states, $q^0 \in Q$ the initial state, $F \subset Q$ the set of final states, and δ the transition function. The language accepted by A is $L(A)$; if $\delta^\star$ denotes the transitive closure of δ, we have $L(A) = \{w, \delta^\star(q^0, w) \in F\}$. A finite-state transducer T is a finite-state automaton with two tapes: an input and an output tape; transitions are labeled with pairs $a : b$, with a in the input alphabet Σ_1, and b in the output alphabet Σ_2. In the rest of this paper, we will only consider the case where $\Sigma_1 = \Sigma_2 = \Sigma$. If T is the transducer, $P_1(T)$ (resp. $P_2(T)$) denotes the finite-state automaton obtained by removing all the output (resp. input) labels.

If M_1 and M_2 are two finite-state automata, we note $M_1 \circ M_2$ the finite-state automaton computing the intersection of $L(A_1)$ and $L(A_2)$. Likewise, if M_1 and M_2 are two finite-state transducers, we note $M_1 \circ M_2$ the transducer computing the letter-to-letter composition of M_1 and M_2.

3.2 Simple analogies

3.2.1 Simple analogical relations

This section introduces an initial definition of an analogy between words and discusses some properties of this definition. As this definition will be generalized in Section 4, analogies considered in this section will be termed 'simple' analogies.

Definition 1 (Simple Analogical relations) *Analogy is a quaternary relationship defined over* $(\Sigma^\star)^4$. *We say that* $(x, y, z, t) \in (\Sigma^\star)^4$ *stand in analogical relationship, noted* $x{:}y{::}z{:}t$ *if and only if* $\exists\{(x_i, y_i, z_i, t_i) \in (\Sigma^\star)^4\}_{i \in [\![1,n]\!]}$ *st.:*

$$x_1 \dots x_n = x,\ y_1 \dots y_n = y,\ z_1 \dots z_n = z,\ t_1 \dots t_n = t$$

and $\forall i \in [\![1, n]\!]$, $(y_i, z_i) \in \{(x_i, t_i), (t_i, x_i)\}$.

The smallest integer n *for which this property holds is termed the* degree *of the analogy.*

This definition captures the intuition that analogies relates words which share a set of common alternating factors. For instance, with this definition we have: $reception : refection :: deceptive : defective$, with $n = 3$ and the following factors: $x_1 = re$, $x_2 = cept$, $x_3 = ion$, $t_1 = de$, $t_2 = fect$, $t_3 = ive$[2]. The degree of an analogical relation (here 3) is a measure of its complexity: the smaller the degree, the better the analogy. This reflects the intuition that good analogies should preserve large portions of the original words.

[2] We have chosen here the linguistically motivated joint segmentation of these four (linguistic) words, but we might as well have chosen $x_1 = r$, $x_2 = ecept$, and accordingly for t_1 and t_2.

Given these definitions, the following properties hold (see also [17]):

$$\forall x \in \Sigma^\star, x : x :: x : x \tag{1}$$

$$\forall x, y \in \Sigma^\star, x : x :: y : y \tag{2}$$

$$\forall x, y, z, t \in \Sigma^\star, x : y :: z : t \Rightarrow z : t :: y : x \tag{3}$$

$$\forall x, y, z, t \in \Sigma^\star, x : y :: z : t \Rightarrow x : z :: y : t \tag{4}$$

which is consistent with the properties listed in 2.2.

An additional property of analogical relationships is the property of symbol inclusion: denoting $\underline{x}$ the set of symbols in Σ which do not occur in x, this property is expressed as:

$$x : y :: z : t \Rightarrow \underline{x} \cup \underline{t} = \underline{y} \cup \underline{z} \tag{5}$$

3.2.2 Analogical equations

The resolution of an analogical equation consists in computing the fourth term of an analogical relationship, given the other three.

Definition 2 ((Simple) analogical equations) *t is a solution to the analogical equation: $x : y :: z :?$ if and only if $x : y :: z : t$.*

An analogical equation have no solution: for instance, given our definition, the equation $abc : def :: ijk :?$ does not have any solution. Condition (5) allows to derive a necessary condition for the analogical equation $x : y :: z :?$ to have a solution: every symbol in x should occur in y or in z. A solution t will in fact contain those symbols in y and z that do not occur in x.

In the general case, analogical equations can also have more than one solution. For instance, $c : ac :: bc :?$ has two equally acceptable solutions: abc and bac. As a consequence of the symbol inclusion constraint, we can note that all the solutions to an analogical equation have the same length.

An analogy solver, or solver for short, defines a partial function from $(\Sigma^\star)^3$ to $2^{\Sigma^\star}$. We will show, in Section 3.4.2, that for simple analogies, this partial function is rational and can be computed using a finite-state transducer. As a first step towards establishing this result, we show in the next section that simple analogical relationships can be expressed using two basic constructions on words and languages, the *shuffle* and the *complementary set* constructions.

3.3 Shuffle and complementary words

3.3.1 Shuffle

The notion of *shuffle* is introduced eg. in [27] as follows:

Definition 3 (Shuffle) *If u and v are two words in $\Sigma^\star$, the shuffle of u and v is the language defined as:*

$$u \bullet v = \{u_1 v_1 u_2 v_2 \ldots u_n v_n,\ \textit{st.}\ u_i, v_i \in \Sigma^\star, u_1 \ldots u_n = u, v_1 \ldots v_n = v\}$$

The shuffle of two words u and v contains all the words w which can be composed using all the symbols in u and v, subject to the condition that if a precedes b in u (or in v), then it must precede b in w. Taking, for instance,

$u = abc$ and $v = def$, the words $abcdef$, $abdefc$, $adbecf$ are in $u \bullet v$; this is not the case with $abefcd$, in which d occurs after, rather than before, e.

The shuffle can also be defined with index sets as:

$$u \bullet v = \{w \in \Sigma^\star \text{ st. } \exists U, V \in \mathcal{I} \text{ with } U + V = I_w, w(U) = u, w(V) = v\}$$

The shuffle operation has the following properties:

$$\begin{aligned} u \bullet \epsilon &= \{u\} && (6)\\ u \bullet v &= v \bullet u \text{ (commutativity)} && (7)\\ (u \bullet v) \bullet w &= u \bullet (v \bullet w) \text{ (associativity)} && (8)\\ u(v \bullet w) &\subset (uv) \bullet w && (9) \end{aligned}$$

The shuffle operation is generalized to languages according to:

$$K \bullet L = \bigcup_{u \in L, v \in L} u \bullet v$$

As noted, for instance in [27], the shuffle of two rational languages is rational. The automaton A, computing $K \bullet L$, is derived from the automata $A_K = (\Sigma, Q_K, q_K^0, F_K, \delta_K)$ and $A_L = (\Sigma, Q_L, q_L^0, F_L, \delta_L)$ recognizing respectively K and L as the product automata $A = (\Sigma, Q_K \times Q_L, (q_K^0, q_L^0), F_K \times F_L, \delta)$, where δ is defined as: $\delta((q_K, q_L), a) = (r_K, r_L)$ if and only if either $\delta_K(q_K, a) = r_K$ and $q_L = r_L$ or $\delta_L(q_L, a) = r_L$ and $q_K = r_K$.

3.3.2 Complementary subwords and complementary sets

The notion of the *complementary subword* is, in some respect, the converse of the shuffle operation, and is defined as follows:

Definition 4 (Complementary subwords and set) *If $u \sqsubset v$, the complementary set of u with respect to v is defined as:*

$$v \backslash u = \{v(I_v \setminus U) \mid U \subset I_v \text{ and } v(U) = u\}$$

If $w \in v\backslash u$, we say that w is a complementary subword *of u in v. If u is not a subword of v, $v\backslash u$ is empty.*

The complementary set of u with respect to v contains what "remains" of v when the symbols in u are removed. For instance, the complementary set of $false$ wrt. $falsehood$ is the singleton $\{hood\}$; the complementary set of ive wrt. $derivative$ is: $\{derativ, dervati, derivat\}$. This operation can be turned into a symmetric binary relationship as follows:

Definition 5 (Complementary relationship) *A word $w \in \Sigma^\star$ defines a binary relationship denoted $\backslash_w$ and defined as: $u \backslash_w v$ if and only if $u \in w\backslash v$.*

This operation can be generalized to languages: if K and L are two languages, the complementary $L\backslash K$ of K with respect to L is the union over words in L of their complementary set with respect to K:

$$L\backslash K = \bigcup_{v \in L, u \in K} v\backslash u$$

The notions of complementary set and shuffle are related through the following property, which is a direct consequence of our definitions.

Proposition 1

$$w \in u \bullet v \Leftrightarrow u \in w \backslash v$$

The concept of complementary subword and sets also yields a finite-state implementation. This derives from the following result:

Proposition 2 *If K and L are rational sets, $L\backslash K$ is also a rational set.*

From $A_L = (\Sigma, Q, q^0, F, \delta)$, the minimal deterministic automaton for L, the finite-state transducer computing complementary subwords with respect to words in L is $T_L = (\Sigma, Q, q^0, F, \delta')$, where δ' is defined as:

$$\begin{cases} \delta'(q, a : \epsilon) = r \text{ if and only if } \delta(q, a) = r \\ \delta'(q, \epsilon : a) = r \text{ if and only if } \delta(q, a) = r \end{cases}$$

It is routine to verify that for any word $x \in \Sigma^\star$, $T_L(x) = L\backslash x$. It directly follows that if K is rational, and A_K is an acceptor for K, then $L\backslash K$ is also rational, computed as the output language of $T_L \circ A_K$. Figure 1 illustrates this construction taking L as the singleton language $L = \{u\}$.

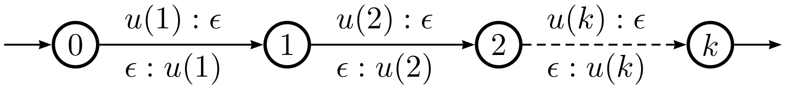

Figure 1: The transducer computing the complementary set of $\{u\}$.

3.4 A Finite-State Analogy Solver

3.4.1 Analogy revisited

The notions of shuffle and complementary sets enable us to provide an alternative characterization of simple analogies between words, based on the following proposition. This proposition gives a necessary and sufficient condition for a simple analogical relation to hold.

Proposition 3

$$x : y :: z : t \Leftrightarrow x \bullet t \cap y \bullet z \neq \emptyset$$

Proof.

Using the notations of definition (1), we can note that if $x : y :: z : t$, then $w = x_1 t_1 \dots x_n t_n$ is a word in $x \bullet t$. As, for each value of i, $x_i t_i$ is either $y_i z_i$ or $z_i y_i$, w also belongs to $y \bullet z$. We conclude that $x \bullet t \cap y \bullet z \neq \emptyset$.

Conversely, assume that w a word in $x \bullet t \cap y \bullet z$. By definition, I_w is partitioned as $I_w = X + T$ and $I_w = Y + Z$, where X, Y, Z, T are index sets such that $w(X) = x$, $w(Y) = y$, $w(Z) = z$ and $w(T) = t$. For all $i \in I_w$, we can have one of the four situations: $i \in X \cup Y$ or $i \in X \cup Z$ or $i \in T \cup Y$ or $i \in T \cup Z$. If $i \in X \cup Y$, we define $x_i = y_i = w_i$, $z_i = t_i = \epsilon$, and likewise for the other cases. The derived joint factorization of x, y, z, t as respectively $x = x_1 \dots x_n$, $y = y_1 \dots y_n$, $z = z_1 \dots z_n$, and $t = t_1 \dots t_n$, satisfies the constraints of definition (1).

The following corollary yields a formal definition of the solutions to an analogical equation:

Proposition 4

$$t \text{ is a solution to } x : y :: z :? \Leftrightarrow t \in y \bullet z \backslash x$$

Proof.

$$\begin{aligned} t \text{ is a solution to } x : y :: z :? &\Leftrightarrow \exists w \in y \bullet z \text{ st. } w \in x \bullet t \\ &\Leftrightarrow \exists w \in y \bullet z \text{ st. } t \in w \backslash x \\ &\Leftrightarrow t \in (y \bullet z) \backslash x \end{aligned}$$

3.4.2 Solving simple analogies on words

Given previous results, a finite-state analogy solver is easily derived. If y and z are in $\Sigma^\star$, we note $M_{y,z} = T_{y \bullet z}$ the finite-state transducer computing the rational relation $\backslash_{(y \bullet z)}$. $M_{y,z}$ is build according to the constructions hereabove detailed for the shuffle and complementary set. The following result is a direct corollary of proposition (4).

Proposition 5 *The solutions to the analogy $x : y :: z :?$ define a rational set; this set is computed as $M_{y,z}(x)$.*

3.5 Complements and perspectives

In this section, we discuss possible extension of this result in the specific context of learning by analogy.

3.5.1 The analogical expansion

The analogical expansion $\mathcal{A}(L)$ of a language L is the set of all words t for which there exists some triple (x, y, z) in L^3 such that $x : y :: z : t$. When L is rational, in particular when M is finite, the computation of $\mathcal{A}(L)$ is performed as: $A_L \circ M_{L,L}$. This computation is a necessary step for performing analogical inference (see Section 1.1). In this context, the finite-state model proposed here has significant benefits: it allows to factorize as much as possible the computations needed during the matching steps and the transfer steps.

It is possible to keep track of the computation of t through $A_L \circ M_{L,L}$ by labelling arcs with 4-uples instead of pairs and thus to output the tuple of words involved in the analogy. The non-determistic machine for $\mathcal{A}(L)$ can be quite large: a naive implementation requires $O(n^3)$ states when A_L has n states. However, this machine can be expanded on-the-fly and does not need to be explicitely stored in memory.

Finally note that the expansion process can be iterated, and we can define the analogical power $\mathcal{A}^i(L)$ of a language L as: $\mathcal{A}(\mathcal{A}(\mathcal{A}...(L)))$. Again, this language can be computed using a finite-state transducer.

3.5.2 Degree, alignment

If x, y, z are words in $\Sigma^\star$, the trace of the computation $t = M_{y,z}(x)$ allows to identify in t two subsets of symbols, corresponding respectively to transitions $\delta(q, \epsilon : y_j)$ and $\delta(q, \epsilon : z_k)$. It can be shown that the number of "discontinuities" in z between symbols from these two subsets corresponds to the notion of degree introduced in Section 3.2 and that the degree is thus computed by a weighted finite-state transducer directly derived from $M_{y,z}$ [33]. Other ways for introducing weights in $M_{x,y}$ can be considered, as will be discussed in Section 5.

This trace also allows to simultaneously *align* x, y, z and t. There are only four types of transitions in $M_{y,z}$, corresponding respectively to: $\delta(q, y(i) : \epsilon)$, $\delta(q, z(j) : \epsilon)$, $\delta(q, \epsilon : y(i))$, and $\delta(q, \epsilon : z(j))$. Each move in $M_{y,z}$ defines an analogical proportion between four symbols: a move $\delta(q, y(i) : \epsilon)$ produces $y(i) : y(i) :: \epsilon : \epsilon$, a move $\delta(q, \epsilon : y(i))$ produces $\epsilon : y(i) :: \epsilon : y(i)$, and likewise for z (see Table 1).

$x =$	w	o	l	ϵ	ϵ	ϵ	f	ϵ	ϵ	ϵ
$y =$	ϵ	ϵ	ϵ	l	e	a	f	ϵ	ϵ	ϵ
$z =$	w	o	l	ϵ	ϵ	ϵ	ϵ	v	e	s
$t =$	ϵ	ϵ	ϵ	l	e	a	ϵ	v	e	s

Table 1: Analogical alignment

3.5.3 Conclusion

In this section, we have provided an initial definition of analogical relationships between words, and demonstrated that this definition is consistent with the traditional use of this notion in linguistics, and that it yields a efficient implemention for the various components involved in an analogical learning system. Extensions of this work to weighted languages have been proposed. In the next Section, we extend this work in another direction, by considering arbitrary analogies between symbols.

4 A general analogical framework

Looking back at definition (1), one can see that the joint factorization of x, y, z, t involve two kinds of factor alignments: those of the (x_i, x_i, t_i, t_i) type and those of the (x_i, t_i, x_i, t_i) type. These alignments of factors can readily be turned into letter-to-letter alignments: this is obvious when x_i and t_i have the same length; otherwise, completing the shortest of the two with the appropriate number of ϵ symbols leads to the desired result.

Such alignments show that our definition of simple analogies between words stems from a (rather trivial) notion of analogy between individual symbols in $\Sigma \cup \{\epsilon\}$, where the only possible analogies take the form $a : a :: b : b$ or $a : b :: a : b$. In this section, our definition is generalized to cases where analogies between symbols take arbitrary forms. This formalism also generalizes the approaches of [15, 20], where the analogy on words is based on string edit distances. We also show that this new definition still yields finite-state analogy solving machines.

We first introduce a handful of new concepts, which are then used to propose a new definition of analogical relationships. We then show how these "generalized" analogies can also be computed with finite-state transducers. We finally discuss possible instanciations of 'arbitrary' analogies at the symbolic level, and study the properties of analogies on words based on the properties of analogies on individual symbols.

4.1 Alignments

In the following, '$-$' will denote a symbol not in Σ and $\overline{\Sigma}$ denotes $\Sigma \cup \{-\}$. $\overline{\phi}$ denotes the function which maps words in $\overline{\Sigma}^*$ to words in $\Sigma^\star$ by erasing occurrences of '$-$'. If x is a word in $\Sigma^\star$, $\overline{\phi}^{-1}(x)$ thus denotes the set of all words w in $\overline{\Sigma}^*$ such that $\overline{\phi}(w) = x$.

Definition 6 (Alignment) *An* alignment *between x and y in $\Sigma^\star$ is a word u in $(\overline{\Sigma} \times \overline{\Sigma})^*$ such that $\overline{\phi}(P_1(u)) = x$ and $\overline{\phi}(P_2(u)) = y$, where P_i denotes the canonical projection on the i-th component of u.*

For instance, $u = (c, g)(a, a)(-, m)(r, e)$ is an alignment between the words *car* et *game* as $\overline{\phi}(P_1(u)) = \overline{\phi}(ca - r) = car$, and likewise for $P_2(u)$. This notion generalizes to n-way alignments:

Definition 7 (n-way alignment) *A n-way* alignment *between n words $x_1 \dots x_n$ in $\Sigma^\star$ is a word u in $(\overline{\Sigma}^n)^*$ such that $\forall i, \overline{\phi}(P_i(u)) = x_i$.*

4.2 Analogical relations and alignments

In the following, we will assume that $\overline{R}$ is a relationship over 4-uples in $\overline{\Sigma}^4$ which specifies analogies over symbols in $\overline{\Sigma}^\star$. $\overline{R}$ is used to define a predicate $\overline{R}^*$ over the set of 4-way alignments according to:

$$\overline{R}^*(u) \text{ iff } \forall i \in I_u, \overline{R}(u(i))$$

This yields a new definition for analogies in $(\Sigma^\star)^4$ according to:

Definition 8 (Generalized Analogical Relationship) $(x, y, z, t) \in \Sigma^\star$ *define a (generalized) analogical relationship, noted $x{:}y{::}z{:}t$ if and only if there exists a 4-way alignment u of x, y, z, t such that $\overline{R}^*(u)$.*

This definition is a proper generalization of the former definition of simple analogies (see definition 1). In fact, if analogies between symbols are defined using R_0, which only contains all the relationships of the form: $\overline{R_0}(a,\ b,\ a,\ b)$ or $\overline{R_0}(a,\ a,\ b,\ b)$, then these two definitions coincide.

Proof. Let t be a solution to the (generalized) analogical equation $x : y :: z :?$. By definition, there is a 4-way alignment u of x, y, z and t such that $\overline{R_0}^*(u)$. This implies that: $\forall i \in I_u, \overline{R_0}(\overline{x}(i), \overline{t}(i), \overline{z}(i), \overline{t}(i))$. We readily derive a joint factorization of x, y, z and t by letting $x_i = \overline{\phi}(\overline{x}(i))$, and likewise for y_i, z_i and t_i. Given the definition of R_0, this alignment satisfies the constraints of simple analogies as stated in definition (1).

Conversely, let t be a solution to the simple analogical equation $x : y :: z :?$. As shown in section 3.4.2, the trace of the computation of t through the finite state solver provides a letter-to-letter alignment possibly involving ϵ symbols (see Table 1). Replacing these with '$-$' directly yields a 4-way alignent u such that $\overline{R_0}^*(u)$.

4.3 Solving generalized analogies

Based on this new definition of analogical relationships, the following solving procedure can be considered:

- Compute $A(x, y, z)$, the set of 3-ways alignments between x, y and z;
- For each $u \in A(x, y, z)$, construct the possible solutions by solving analogies on a per symbol basis. Formally, if for all $i \in I_u$, the 3-uple $u(i)$ can be completed with $\overline{t}(i)$ such that $R(u(i), \overline{t}(i))$, then output $\overline{\phi}(\overline{t})$ as a solution.

These computations are easily implemented with finite-state machines. We first introduce the transducer G which computes $\overline{\phi}$, displayed on Figure 2. G^{-1} is the transducer computing the inverse function, obtained by exchanging the input and output tapes.

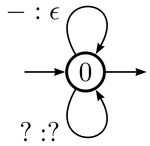

Figure 2: The transducer G computing $\overline{\phi}$.
The transition labeled $? :?$ *represents all possible transitions* $x : x$, *for* $x \in \Sigma$.

Let T_x, T_y and T_z be the three transducers displayed on Figure 3 and A_s be $s \circ G^{-1} \circ T_s$, for $s = x, y, z$. The output language of A_x (resp. A_y, A_z) is the set of all alignments between x (resp. y and z) and any two other words; words recognized by $P_2(A(x))$ are such that symbols in $\overline{x}$ (resp. $\overline{y}$ and $\overline{z}$) in the alignment occurring in the first (resp. second and third) component in the triples (see Figure 4).

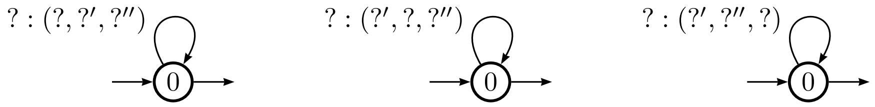

Figure 3: The transducers T_x, T_y, T_z.
The transition labelled (?:(?,?',?")) represents all the transitions of the form $(a : (a, b, c))$ *for* (a, b, c) *in* $\overline{\Sigma}^3$.

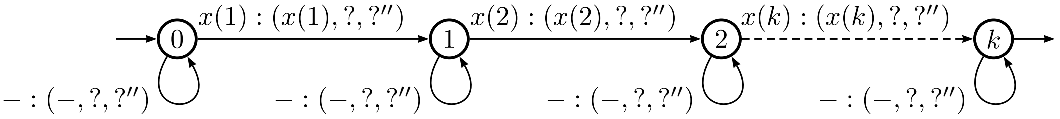

Figure 4: The transducer A_x.

The set of all alignments between x, y and z is hence computed as the intersection $P_2(A_x) \circ P_2(A_y) \circ P_2(A_z) = A(x, y, z)$.

The second step of the algorithm, consisting of the resolution of analogies over letters, is performed as $A(x, y, z) \circ S$, where S is the atomic solver displayed on Figure 5.

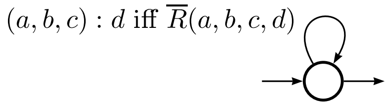

Figure 5: The atomic solver S.

Finally, the set of solutions is computed as $\overline{\phi}(T)$, where T is the output language of $A(x, y, z) \circ S$. Overall, we thus compute the set of solutions to $x : y :: z :?$ as:

$$G \circ P_2(P_2(x \circ G^{-1} \circ T_x) \circ P_2(y \circ G^{-1} \circ T_y) \circ P_2(z \circ G^{-1} \circ T_z) \circ S)$$

The resulting machine has a number of states equal to $|x| \times |y| \times |z|$ .The number of arcs depends on the number of actual transitions in S.

4.4 Analogies over a finite set

Previous sections have shown that the notion of generalized analogies entirely relies on the definition of a relation R over 4-uples of symbols over a finite alphabet. In this section, we first study the properties of the analogies in $\overline{\Sigma^\star}$ based on the properties of $\overline{R}$, and discuss the relevance of such extensions by providing various application contexts.

4.4.1 Basic properties of $\overline{R}$

First of all, it seems reasonable to expect $\overline{R}$ to satisfy the conditions discussed in Section 2.2. If this is the case for the properties "symmetry of as" and "exchange of means", then it is routine to verify that these properties will carry out to analogies on words.

This is not the case with the property of determinism of $\overline{R}$, which does not necessarily imply the determinism of the analogy between words. In fact, it has been shown in Section 3 that this was already the case with 'simple' analogies.

In our system, the symbol '$-$' plays a distinguished part, as it allows to compute alignements for words of different length. A corollary is that if the definition of $\overline{R}$ never involves this symbol, analogies will only exist between words having the same length, which would be a serious limitation of our model. Conversely, it seems reasonable to prohibit analogies of the form: $\overline{R}(-, -, -x)$. This is because the output languages of A_x, A_y and A_z contain words of arbitrary length, due to the possible insertions of an arbitrary number of '$-$' symbols. The possibility to solve $\overline{R}(-, -, -x)$, with $x \in \Sigma$ would directly yield an infinite number of solutions.

4.4.2 Symbols as feature vectors

A useful and common representation in computational linguistics (and, more generally, in many areas of Artificial Intelligence) is to model linguistic units as vector of features[3]. A typical example, in computational phonology, is the representation of the minimal units of the sound system as vectors of binary (distinctive) features, each feature denoting a minimal opposition between at least two sounds. In this system, it is classical, to distinguish the phonemes /b/ and /d/ based on the value of the 'voicing' feature: /b/ is the voiced counterpart of the unvoiced labial stop /p/. The same feature also discriminates other pairs of consonants such as: /d/ and /t/, /g/ and /k/, /v/ and /f/... Similarly, /b/ and /d/ are both voiced stops, which are distinguished based on the value of features describing the place (labial vs. dental), rather than the manner, of articulation.

Based on such feature-based description, an analogical relationship R over the inventory of phonemes is readily deduced as $R(a, b, c, d)$ if the distinctive opposition between a and b is the same as the opposition between c and d. This definition satisfies the minimal requirements described above, and allows to recognise analogies between linear phonological representations such as, for instance: $/bik/ : /pig/ :: /duf/ : /tuv/$, where the same alternation between voiced and unvoiced consonants is observed in both pairs. This definition of R has an additional interest: each analogy $R(a, b, c, d)$ can be weighted with an integer value, based on the number of differences between the descriptions of a and b [21]. We will return to this point in Section 5.

Other feature-based representations of linguistic units, which can easily be turned into analogies on a finite set, include representation of morpho-syntactic properties (one feature for the main category, one for the number, the gender...), or feature-based representations of the semantic content of lexical units.

4.4.3 Structured sets of symbols

Another potentially interesting situation corresponds to the case where Σ is fully equipped with an internal operation, $\oplus$, providing Σ with an algebraic structure. It is then worth trying to figure out the conditions on $\oplus$ which are compatible with the usual properties of analogical relationships.

Let us first recall the definition of an analogy in a vector space. In a vector space, the analogy $\overrightarrow{A} : \overrightarrow{B} :: \overrightarrow{C} : \overrightarrow{X}$ is solved by taking $\overrightarrow{X}$ as the fourth summit of the parallelogram built on $\overrightarrow{A}$, $\overrightarrow{B}$ and $\overrightarrow{C}$. A parallelogram being defined as:

$$\overrightarrow{A} + \overrightarrow{X} = \overrightarrow{B} + \overrightarrow{C}$$

it seems quite natural to take this equation as a starting point for studying the properties of $\oplus$. An analogy over $(\Sigma, \oplus)$ is thus simply defined as:

$$a : b :: c : x \Leftrightarrow a \oplus x = b \oplus c \tag{10}$$

This definition is compatible with trivial analogies of the kind: $a : a :: b : b$.

Properties of analogy We have given, in Section 2, a list of reasonnable requirements for an analogical relationship. Let us rewrite them with the operator $\oplus$. For each axiom, we identify the corresponding requirement on $\oplus$, allowing us to determine more precisely what properties $\oplus$ must verify.

[3]We will only consider here the case where the feature set is finite, each feature taking its values from a finite set. For a more general notion of analogies which also applies to recursive feature structures such as the ones typically used in unification grammars, see [29].

Exchange of the means is expressed as follows:

$$a : b :: c : d \Rightarrow a : c :: b : d$$

which translates into:

$$(a \oplus d = b \oplus c) \Rightarrow (a \oplus d = c \oplus b)$$

This condition directly implies that $\oplus$ must be commutative: since $a : a :: b : b$, this condition directly yields: $\forall a, b, (a \oplus b = b \oplus a)$. Moreover, commutativity of $\oplus$ also entails the property of symmetry, since symmetry, expressed as:

$$(a : b :: c : d) \Rightarrow (c : d :: a : b)$$

translates into:

$$(a \oplus d = b \oplus c) \Rightarrow (c \oplus b = d \oplus a)$$

which is true if $\oplus$ is commutative.

Determinism is expressed as follows:

$$(a : a :: c : x) \Rightarrow (x = c)$$

rewritten as:

$$(a \oplus x = a \oplus c) \Rightarrow (x = c)$$

This condition implies that $\oplus$ must be a left regular operator on Σ. Since commutativity of $\oplus$ is also required, this condition entails that $\oplus$ must be regular.

The alphabet as a finite group These properties are obviously satisfied if we take $(\Sigma, \oplus)$ as an Abelian (commutative) group[4]. We thus study the case of the cyclic finite group. The table describing $\oplus$ in the case $|\Sigma =| 6$ is given in Table 4.4.3. With this definition, it is possible to solve every analogical equation on letters.

$\oplus$	a	b	c	d	e	f
a	a	b	c	d	e	f
b	b	c	d	e	f	a
c	c	d	e	f	a	b
d	d	e	f	a	b	c
e	e	f	a	b	c	d
f	f	a	b	c	d	e

Table 2: The table for $\oplus$ in the cyclic finite group $\mathcal{G}_6$. a is the identity element.

For instance, the unique solution to the analogical equation: $a : d :: c :?$ is directly read in Table 4.4.3 as $? = f$. Using $\oplus$ to define the analogical relationship R over Σ, and further assuming a total ordering of Σ, we are in a

[4] $(G, \oplus)$ is a group if and only if:

- $\oplus$ is an internal operation
- $\oplus$ is associative $(a \oplus b) \oplus c = a \oplus (b \oplus c)$
- Σ contains an identity element for $\oplus$, denoted 0:
- every element x of G has a symmetric $\bar{x}$: $\forall x \in G, \exists \bar{x} \mid x \oplus \bar{x} = \bar{x} \oplus x = 0$

position to solve analogical equations such as: $abc : abd :: cde :?$ $(? = cdf)$ or even $bcd : abe :: fec :?$ $(? = edd)$, where the precedence/successor relationship over symbols is implicitely taken into account.

Thus far, we have used $\oplus$ to define an analogical relationship R between 4-uples of symbols in Σ. Assuming, for instance, that $\overline{R}$ coincides with R over Σ, and is completed with: $\forall a \in \Sigma, \overline{R}(a, a, -, -)$ and $\overline{R}(a, -, a, -)$, analogies such as $abc : ad :: cbe :?$ can also be solved $(? = cf)$. Other definitions of $\overline{R}$ can also be entertained: it is, for instance, possible to deduce $\overline{R}$ directly from a definition of $\oplus$ over $\overline{\Sigma}$. These alternative models will potentially yield different operational instantations of analogical relationships.

4.5 Conclusion and complements

In this section, we have proposed a generalized model of analogical relationships for words based on the definition of an analogical relationship over $\overline{\Sigma}$. We have shown that, in this model, the resolution of analogical equations could still be carried out using finite-state transducers. We have also discussed the relevance of this model for various application domains.

This preliminary work opens the door for many extensions: for instance, although we have assumed that an analogical relationship involves four terms, our definition and resolution procedure directly generalize to the case where $\overline{R}$ is an arbitrary n-ary relationship. The applicability of this generalization however remains to be assessed.

A more complex situation can finally be taken into account in this model, corresponding to *multi-level* analogies. Again, natural language applications abounds in examples of this situation: for instance, analogies between (linguistic) sentences can be seen as analogies over sequences of words, analogies between the basic units (words) at this first level being decomposed into analogies between between indivual phonemes or letters. However, the computational complexity of such models remains to be carefully studied, given that the number of units over which the basic analogical relationship is defined (here, linguistic words) is not necessarily finite.

5 Ranking analogies

In the previous sections, we have proposed various possible definitions of an analogy between words, and demonstrated that these definitions were likely to yield analogical solvers taking the form of finite-state transducers. Depending on the definition of the underlying analogical relationship between symbols in Σ, the number of solutions to an equation can be quite large. One would then like to be able to specify some kind of quantitative measure to each analogy, allowing to rank the various solutions to a given equation. In this section, we introduce the notion of the *cost of an analogy*, and consider possible definitions and uses of this new notion.

Another important concern is the complexity of the resolution procedure: analogical inference, as introduced in Section 1.1, requires to solve repeatedly a large number of analogical equations, both during the matching and the transfer step. We thus present a fast heuristic resolution procedure, suitable for situations where the number of candidate analogs is sufficiently large to dispense with the exploration of the complete set of candidates. This procedure is based on a distance between individual symbols, from which the cost of an analogy is derived. In this context, we show that our algorithm computes an approximation of the lower cost solution.

5.1 The cost of an analogy

As previously discussed, each analogical relationship over words is decomposed into a series of atomic analogies over symbols in $\overline{\Sigma}$. Assuming that atomic analogies are weighted according to some positive cost function c from $\overline{\Sigma}^4$ into $\mathbb{R}^+$, the cost c^* of a 4-way alignment u is defined as:

$$c^*(u) = \sum_i c(u(i)) \tag{11}$$

An analogy $x : y :: z : t$ being supported by at least one 4-way alignment u such that $\overline{R}^*(u)$, the cost of an analogy $x : y :: z : t$ is defined:

$$C(x : y :: z : t) = \min_{u \in A(x,y,z,t) \overline{R}^*(u)} c^*(u) \tag{12}$$

where $A(x, y, z, t)$ is the set of all 4-way alignments between these words.

In this setting, the resolution of the analogical equation $x : y :: z :?$ consists in finding the word(s) t^* such that:

$$t^* = arg \min_{t \in \Sigma^\star} C(x : y :: z : t)$$

This definition covers a large number of possible implementations. In any case, it is reasonnable to impose conditions such as:

- $c(a, a, a, a) = 0$
- $\forall a, b \in \Sigma, c(a, b, a, b) = c(a, a, b, b) = c(b, b, a, a)$
- $\forall a, b, c \in \Sigma c(a, a, b, b) < c(a, a, b, c)$ if $b \neq c$
- $\forall a, b, c \in \Sigma c(a, b, b, b) \leq c(a, -, a, -) + c(-, b, -, b)$

in order to ensure that low cost analogies still match the intuition that the best analogy is the identity $a : a :: a : a$.

This also offer a potential framework for defining *probabistic analogical models*. Assuming that c satisfies the condition:

$$\forall a, b, c, \in \Sigma, \sum_{x \in \overline{\Sigma}} \exp(-c(a, b, c, x)) = 1$$

allows to interpret $\exp(-c(a, b, c, .))$ as a probability distribution $P_{abc}()$. Contextual dependencies between successive 4-uples in an alignment are readily introduced if one makes $c(u(i))$ depend from $u(i-1)$: this idea is in fact implicitely present in our definition of the degree of an analogical relationship (see Section 3).

In any case, this definition of the cost of an analogy remains compatible with a finite-state resolution procedure, and preserves most of its benefits. In fact, it is sufficient to turn the atomic solver S introduced in Section 4.3 into a weighted finite-state transducer. Finding the best solution to an analogical equation thus amounts to the search of a minimal cost path in the analogy solving transducer, and can be performed using efficient exact or heuristic search procedures [25].

In the next section, we consider the case where c is based on the definition of a distance between atomic symbols in $\overline{\Sigma}$, and derive a heuristic algorithm for finding the best solution to an analogical equation.

5.2 A heuristic analogical solver

5.2.1 The edit distance between strings

As described in subsection 4.2, an alignment between x and y induces a one-to-one mapping between symbols in $\overline{x}$ and $\overline{y}$, which can be interpreted as a series of elementary operations, called the *edit operations* [30], which rewrite x into y:

- an alignment $(x(i), -)$ denotes the *deletion* of $x(i)$;
- an alignment $(-, y(j))$ denotes the *insertion* of $y(j)$;
- an alignment $(x(i), y(j))$ denotes the *subsitution* of $x(i)$ by $y(j)$.

For instance, the alignment $u = (a,a)(b,c)(-,d)(e,e)(f,-)$ can be read as a series of 3 substitutions, one deletion and one insertion, as displayed in the following table:

$$\begin{array}{lcccccc} P_1(u) = \overline{x} & = & a & b & - & e & f \\ & & | & | & | & | & | \\ P_2(u) = \overline{y} & = & a & c & d & e & - \end{array}$$

Assuming that a cost is assigned to each edit operation, it is possible to compute a cost for any alignment. Let δ be a positive function on $\overline{\Sigma} \times \overline{\Sigma} \longmapsto \mathbb{R}$, giving the cost of each edit operation. The cost of an alignment is defined as follows:

Definition 9 *The* cost $\delta^*(u)$ *of an alignment u is the sum of the costs of the edit operations composing the alignment, i.e.:*

$$\delta^*(u) = \sum_{i \in I_u} \delta(u(i))$$

For instance, if $\forall a, b \in \overline{\Sigma}, \quad \delta(a,b) = 1 \text{ if } a \neq b, \text{ and } \delta(a,b) = 0 \text{ if } a = b$, the alignment: $u = (a,a)(b,c)(-,d)(e,e)(f,-)$ has a cost:

$$\delta(a,a) + \delta(b,c) + \delta(-,d) + \delta(e,e) + \delta(f,-) = 0 + 1 + 1 + 0 + 1 = 3$$

If δ is a distance[5]in $\overline{\Sigma}$, the *edit distance* between words in $\Sigma^\star$ is defined as follows (see eg. [2]):

[5]A distance δ is a mapping $E \times E \longmapsto \mathbb{R}^+$ such that:

- $\forall x, y \in E, \delta(x,y) \geq 0$ (positivity);
- $\delta(x,y) = 0 \Leftrightarrow x = y$;
- $\forall x, y \in E, \delta(x,y) = \delta(y,x)$ (symmetry);
- $\forall x, y, z \in E, \delta(x,y) \leq \delta(x,z) + \delta(z,y)$ (triangular inequality).

Definition 10 *The* edit distance $d(x, y)$ *between two words* x *and* y *is the cost of the alignment between* x *and* y *which has the lowest cost, i.e.:*

$$d(x, y) = \min_{u \in A(x,y)} \delta^*(u)$$

where $A(x, y)$ denotes the set of alignments between x and y.

A minimal cost alignment is termed *optimal*. The optimal alignment may not be unique. The computation of the optimal alignment can be performed using dynamic programming procedures ([30, 28]).

5.2.2 Relating analogical costs and edit distance

Assuming δ is a distance, a cost function for atomic alignments in $\overline{\Sigma}^4$ is defined as:

$$c(a, b, c, d) = \frac{1}{2}(\delta(a, b) + \delta(a, c) + \delta(b, d) + \delta(c, d)) \tag{13}$$

This cost function has several desirable properties. In particular we have:

- $c(a, b, c, d) = 0 \Leftrightarrow a = b = c = d$
- $c(a, b, c, d) = c(a, c, b, d) = c(c, d, a, b)$
- $\forall c \neq b, c(a, a, b, b) \leq c(a, a, b, c)$, by application of the triangular inequality

These properties are in line with the intuition that the good analogies on words tend to use atomic analogies of the kind $a : a :: a : a$, $a : b :: a : b$ or $a : a :: b : b$.

We will additionnally require the following property, which links more closely the definition of the distance δ and the possible atomic analogies induced by $\overline{R}$:

$$\overline{R}(a, b, c, d) \Rightarrow \delta(a, b) = \delta(c, d) \text{ and } \delta(a, c) = \delta(b, d) \tag{14}$$

In this case, $c(a, b, c, d)$ simplifies as $\delta(a, b)+\delta(a, c)$. This property is satisfied, irrespective of δ, when we consider only simple atomic analogies $a : a :: b : b$ or $a : b :: a : b$. As we will see later, this property is also true when we consider specific distances over symbols in the Abelian finite group.

5.2.3 A heuristic analogical solver

In this section, we propose a heuristic approach for solving analogical equations on words, which aims at finding the minimal cost analogy, when this cost is based on distance between symbols. This algorithm can be considered as a generalization of the algorithm originally introduced in [15] which only considers the usual edit distance and basic analogies between letters. The general idea is to reduce the search space by examining only a (small) subset of the possible 3-ways aligments between x, y, z, when the exact approach detailed in Section 4.3 requires to consider all of them.

Given the equation $x : y :: z :?$, the best solution is computed as:

$$t^* = arg \min_{t \in \Sigma^*} C(x : y :: z : t) \tag{15}$$

Denoting $U = \{u \in A(x,y,z,t)\overline{R}^*(u)\}$ the set of all alignments supporting the analogy $x : y :: z : t$, this rewrites as:

$$t^* = arg \min_{t \in \Sigma^\star}(\min_{u \in U} c(u)) \quad (16)$$

$$= arg \min_{t \in \Sigma^\star}(\min_{u \in U} \sum_i c(u(i))) \quad (17)$$

$$= arg \min_{t \in \Sigma^\star}(\min_{u \in U} \sum_i \delta(\overline{x}(i), \overline{y}(i)) + \delta(\overline{x}(i), \overline{z}(i))) \quad (18)$$

The summation in (18) contains two terms: $\sum_i \delta(\overline{x}(i), \overline{y}(i))$ and $\sum_i \delta(\overline{x}(i), \overline{z}(i))$. The main point of our approach consists in optimizing these two quantities independantly, yielding an alignment between x and y in the one hand, and x and z in the other hand; then to merge these alignments in such a way that one or several solutions t may be computed.

The first part of this program consists in computing the optimal alignments between x and y (denoted u_1), and between x and z (denoted u_2). This can be done using the classical algorithms for computing edit distances between words [30].

The resulting pair of alignments can be turned into a 3-way alignment between x, y and z by inserting extra pairs $(-,-)$ in u_1 and/or u_2 in such a way that the first component of u_1 and u_2 are made equal. This amounts to searching only a small subset of the intersection $P_2(A_x) \circ P_2(A_y) \circ P_2(A_z)$ introduced in the general resolution procedure.

This operation is the second step of the algorithm and can be performed in linear time, as detailled in algorithm 1.

Algorithm 1 Aligning alignments

begin
$//insert(u, i, a)$ inserts symbol a in word u at index i.
$i_1 \leftarrow 1$; $i_2 \leftarrow 1$;
while ($i_1 \leq |u_1|$ or $i_2 \leq |u_2|$) **do**
  if ($\overline{x}_1$ = '-' and $\overline{x}_2 \neq$ '-') **then**
    $insert(u_2, i_2, (-,-))$;
  end if
  if ($\overline{x}_1 \neq$'-' and $\overline{x}_2$ ='-') **then**
    $insert(u_1, (-,-))$;
  end if
  if ($\overline{x}_1$ = '-' and $\overline{x}_2$ = '-') **then**
    $insert(u_1, i_1, (-,-))$ or $insert(u_2, i_2, (-,-))$;
  end if
  $i_1 \leftarrow i_1 + 1; i_2 \leftarrow i_2 + 1$;
end while
end

The main observation regarding this algorithm is that it is non-deterministic in nature: cases when both the optimal alignments of x and y and of x and z both 'insert' symbols in x at the same position, meaning that '-' occurs simultaneously in $\overline{x}_1$ and $\overline{x}_2$ at the same position, yield atomic equations of the kind $- : b :: c :?$. In most cases, these cannot be solved and have to be replaced with the two equations: $- : b :: - :?$ and $- : - :: c :?$. These two replacements can be performed in any order, which makes the algorithm non-deterministic. Various proposals have been made to make this choice deterministic: see eg. [15] or [5]. The notion of degree introduced earlier (see Section 3.2.1) can also be used to make this decision. In any case, even if this part of the resolution algorithm is made deterministic, the overall solver might still yield several solutions: firstly because the optimal alignments are not necessarily unique, second because of the non-determinism of $\overline{R}$ itself. The output of this algorithm is a 3-way alignment u between x, y and z.

The final step of the resolution procedure simply consists in solving atomic analogies for each triple in u. Again, depending on the definition of $\overline{R}$, this step may yield one, several or even no solution at all.

An example Let $x = wolf$, $y = wolves$, $z = leaf$, and u_1 and u_2 be the results of the optimal alignments between x and y, and x and z.

$$u_1 = \begin{pmatrix} w & o & l & - & f & - \\ | & | & | & | & | & | \\ w & o & l & v & e & s \end{pmatrix} \quad u_2 = \begin{pmatrix} w & o & l & - & f \\ | & | & | & | & | \\ - & l & e & a & f \end{pmatrix}$$

$P_1(u_1)$ and $P_1(u_2)$ having a common prefix of length 3 (wol), their prefixes can directly be merged into a 3-way alignment $(w, w, -)(o, o, l)(l, l, e)$. The next iteration is a case where an (non)-deterministic insertion has to take place. Assuming it takes place in u_1, the output is first completed with $(-, -, a)$, then with $(-, v, -)$. The two last iterations match the 'f' symbol, then yield an insertion in u_2. Overall the resulting 3-way alignment is:

$$(w, w, -)(o, o, l)(l, l, e)(-, -, a)(-, v, -)(f, e, f)(-, s, -).$$

Assuming $\overline{R}$ contains $\overline{R_0}$, a solution is finally found as: $t = \overline{\phi}(-leaves) = leaves$.

5.2.4 Application: the additive cyclic group

The algorithm presented above works well in the case of the traditional edit distance and trivial analogies over $\overline{\Sigma}$. Another potentially interesting situation is when Σ, equipped with an internal composition law, is homomorphic to the cyclic group (see Section 4.4).

To make the heuristic approach applicable, a distance δ over Σ has to be defined such that, given the definition of atomic analogies induced by $\oplus$, we have:

$$\forall a, b, c, d \in \Sigma, a : b :: c : d \Rightarrow \delta(a, b) = \delta(c, d) \tag{19}$$

Assuming Σ has n elements, and denoting 0 the identity element, we first introduce $\forall 0 \geq i \leq n, d_i = \delta(0, i)$, with $d_0 = 0$. Condition (19) yields $\forall a, b \in \Sigma, \delta(a, b) = \delta(0, b \oplus \overline{a})$. Symmetry of δ also implies that: $\forall i, d_i = d_{n-i}$: δ is thus entirely defined by $\lfloor \frac{n-1}{2} \rfloor$ values. The last condition is imposed by the triangular inequality, which yields:

$$\min_{i,j,i \neq j} d_i + d_j \geq \max_i d_i.$$

Conversely, any set of $\lfloor \frac{n-1}{2} \rfloor$ values satisfying these constraints will define a distance over Σ. [5] suggests to consider $\{d_i\}$ as a set of evenly spaced points on a circle, but this is by no means the only solution.

Based on a definition of a distance δ over symbols in Σ, the heuristic algorithm presented in Section 5.2 can directly be used, and provides a solution for every analogical solution.

5.3 Conclusion

In this section, we have introduced the notion of the cost of an analogy and discussed ways in which such a cost function could be introduced. In this context, the resolution of an analogical proportion consists in finding the minimal cost path in a weigthed finite-state transducer, for which efficient exact or heuristic procedures exist. We have also discussed one possible definition of this cost, based on the notion of edit distance. To deal with these cases, we have finally proposed a heuristic resolution procedure, the complexity of which is lesser than the general resolution algorithm.

6 Related work

6.1 Solving difficult analogical problems on words

The Copycat model, introduced in [23], aims at providing a computational model of the way humans solve analogical problems involving letter strings. The main goal of this work is to built a system replicating the human preference for analogies involving abstract (conceptual) similarities over purely formal ones. Consider for instance the problem: $abc : abd :: ijk :?$; the solution ijk, which involves the conceptual notion of successorship, makes a better solution than the "simpler" ijd or abd. Conceptual similarities are not based on any predefined measure; rather they dynamically emerge under the pressure brought by the perceptual analysis of the situation. These effects are achieved in a distributed architecture, involving a static network of concept (the *Slipnet*), through which activation is likely to spread between adjacent nodes, and simple software agents, referred to as *codelets*. These codelets form coalitions which compete to produce a consistent conceptual analysis of the strings involved in the problem at hand. Promising analyses contribute to reinforce the activation of the corresponding concepts, which in turn improves the ressources allocated to the associated codelets. Starting from a situation where codelets randomly explore the space of possible perceptual structures, this dynamic system progressively reaches a stable state, through a global *temperature* parameter, the slow decrease of which makes the codelets more and more deterministic. Based on these principles, Copycat is able to solve remarkably sophisticated analogical equations, such as $abc : abd :: mrrjjj :?$. In this case, the computation of the solution $mrrjjjj$ requires to establish a mapping between the successorship relationship, used to produce an analysis of the first pair abc : abd, and the iteration operator, involved in the analysis of $mrrjjj$: in this problem, changing c to its successor d is thus viewed as analogous to iterating j four times instead of 3.

Addressing similar types of problems, [3] introduces a general algebraic approach to the problem of solving analogical equations on sequential patterns, focusing especially on words. A series of domain-independant and domain-dependant primitive operators, such as iteration, symetry, concatenation, successor... allows to *parse* any complex pattern as the result of the application of one or several operators to simpler patterns. *abc*, for instance, results from the combination of the iteration ($Iter$) and the successor ($Succ$) operators, and is parsed as: $Iter(a, succ, 3)$, which reads: "concatenate a, its successor and the successor of its successor". In this model, a pattern may be ambiguous, meaning that it can be parsed in more than one way. Two patterns x and y are perceived *as similar* if they can be parsed with the same combination of operators. In this context, an analogical proportion exists between x y z and t if and only if (i) x and y on the one hand, and z and t in the other hand are perceived as similar (as explained above), and if there exists an isomorphism between the operators generating x and y, and the operators generating z and t. This basic model is completed with a quantitative measure (the "information load") of the complexity of a similarity between x and y, based on the number of operators it involves. This measure also allows to rank analogical proportions, and is used to solve analogical equations: the solution to $x : y :: z :?$ will be chosen so as to minimize the joint information load. Finally, Dastani and his co-authors present a heuristic algorithmical resolution procedure, which exploits the representation of parses of a pattern as an acyclic graph.

The two models presented in this section display a capacity to solve analogical equations on words which goes far beyond the capacities of our own model. As we see it, this is as much an advantage as a weakness, for at least two reasons. Firstly, they use the word domain as a mere illustration of a general model of analogical reasoning, and consider conceptual relationship between letters or groups of letters that are of little bearing for our linguistic purposes: the concept of successorship, for instance, seem to play little role in the perception of actual linguistic words. Conversely, it is difficult to foresee how the distributional properties of letters (or sounds, or words...), which are of significant importance to analyse linguistic data, could be integrated into such purely symbolic approaches. Second, the analogical resolution procedure they consider has a computational complexity which makes their integration in our analogical learning model quite unrealistic.

6.2 Two solutions from Yves Lepage

The modeling of analogical proportions between words has so far received little attention. Most relevant however to our discussion is the work of Yves Lepage, especially [15] and [17], presented in full details in [18].

[15] details the implementation of an analogy solver for words, based on a generalization of algorithms computing the edit distance between strings. Lepage first introduces a dissimilarity measure $pdist(u, v)$ between strings u and v. In essence, $pdist(u, v)$ is computed as the traditional edit distance [30], *using a null insertion cost*. Given the equation $x : y :: z :?$, this algorithm proceeds as follows:

- (1) compute, using dynamic programing techniques, $pdist(x, y)$ and $pdist(x, z)$. This procedure induces an optimal alignment between x and y on the one hand, and x and z on the other hand.
- (2) traverse *simultaneously* the alignments produced in step (1.), while

writing on an output tape the symbols in y or z which are not matched in x. Alternative traversals are selected using a heuristic, which basically ensures that the traversal remains "close" to the diagonal. This procedure is prone to failure, corresponding to the processing of a symbol in x which is neither matched in y, nor in z.

The first step of this algorithm is reminiscent of the first step of the algorithm presented in Section 5.2 and produces one (optimal) joint alignment of x and y on the one hand, and of x and z in the other hand. The second step of this procedure is similar to ours, and finds an optimal merge of these alignments, while at the same time solving analogies between individual letters. In Lepage's model, only trivial analogies induced by the $\overline{R_0}$ relationship are considered. Like the heuristic resolution procedure presented in section 5.2.3, this algorithm can be viewed as a way to select some solutions amongst the ones that are found by the more general resolution procedure presented in Section 4. Considering the optimal alignments induced by $pdist()$, instead of $dist()$, is a way to enforce alignements (a, a) (copy) and $(-, a)$ (insertion) over (a, b) (substitution) or $(a, -)$ deletion. As a result, Lepage's algorithm is less prone to failure: $abba : aacc :: bb :?$ has one solution (cc), whereas the heuristic approach presented in section 5.2.3 does not find any.

[17] goes one step further, and proposes a more powerful model of analogy between words with the potential to generate non-rational languages, using solely analogical proportions. In this work, analogical proportions between words are defined based on combinatorial axioms rather than on an algorithmic procedure. These axioms correspond to properties (3) and (4). Lepage then considers the effect of the concatenation operation on analogical relations and states the following axiom:

Axiom 1 (Concatenation) *Let Σ a finite alphabet and Σ_1 and Σ_2 two disjoint subsets of Σ; then for all $(x_1, y_1, z_1, t_1) \in \Sigma_1^4$, $(x_2, y_2, z_2, t_2) \in \Sigma_2^4$ such that both $x_1 : y_1 :: z_1 : t_1$ and $x_2 : y_2 :: z_2 : t_2$, we also have:*

$$\begin{cases} x_1x_2 : y_1y_2 :: z_1z_2 : t_1t_2 \\ x_1x_2 : y_1y_2 :: z_2z_1 : t_2t_1 \\ x_1x_2 : y_2y_1 :: z_1z_2 : t_2t_1 \end{cases}$$

The next step is to define what Lepage calls *language of analogical strings*. To this end, the following generative model is introduced: given M, a finite set of pairs from $\Sigma^\star$, the derivation relationship $\vdash_M$ over $\Sigma^\star$ is defined as:

$$x \vdash_M y \text{ iff } \exists (z, t) \in M \text{ such that } x : y :: z : t.$$

Denoting $\vdash_M^\star$ the transitive closure of $\vdash_M$, the langage of analogical strings generated from the finite set A is defined as:

$$L_M(A) = A \cup \{y, \exists x \in A, x \vdash_M^\star y\}.$$

Given these definitions, Lepage then claims that if we take $M = \{ab\}$ and $M = \{ab, aabb\}$, we must have: $L_M(A) = \{a^nb^n\}$. To prove this result, two new sets of analogical patterns are introduced, which hold for every value of n: $a : aa :: a^{n-1} : a^n$ and $b : bb :: b^{n-1} : b^n$. The concatenation axiom (axiom 1) thus allows to derive the following analogy: $a^{n-1}b^{n-1} : a^nb^n :: ab : aabb$, from which it derives that if ab is in $L_M(A)$, then also $a^n b^n$ is in $L_M(A)$ for all n. Note that another concatenation rule could also apply: $b^{n-1}a^{n-1} : b^n a^n :: ab : aabb$: however, as ba is not in A, this rule can not be used and no word in $\{b^n a^n, n > 0\}$ is generated. From there, it is legitimate to assume that, using the concatenation axiom only, we will only produce words in $a^n b^n$. It is however unclear why one should restrict oneself to these kinds of "compound" analogies: in fact, solving $aabb :? :: ab : aabb$, we find that $aababb$, which does not belong to $\{a^n b^n\}$, can also be derived[6] from ab. From there, many more words not in $\{a^n b^n\}$ can also be generated. The additional restriction to use solely "compound" analogies, *based on a set of primitive patterns holding independantly for disjoint subsets of the alphabet*, makes the model more constrained and allows to generate non-rational languages. This restriction however requires a more complex computational model than the one used throughout this paper.

7 Conclusion

In this paper, we have proposed a sound definition of an analogy between linear strings of symbols, which solely relies on the definition of valid analogies between individual symbols. We have then proved that it yielded effective resolution procedures, based on finite-state transducers. We have also discussed possible ways to integrate these procedures into analogical learning devices.

To address the intrinsic non-determinism of such analogy-solving machines, we have successively explored means to rank competing analogies, based on a measure of the quality of individual analogies, and proposed a fast heuristic approach aimed at computing the lowest cost solution(s) to an analogical equation.

The work reported here is only a first step towards validating our initial claim that analogical learning has the potential to provide effective means for solving a wide number of NLP tasks. Various perspectives, some of which have already been mentionned in the text, remain to be explored:

- from a computational perspective, our model, based on non-deterministic finite-state transducers, leaves some room for improvement. In particular, procedures for filtering spurious ambiguities, and for speeding up the resolution procedure in the presence of external domain knowledge remain to be explored;

- probabilistic analogical models introduced in Section 5.1 are probably the "right" answer to non-determinism, and can be considered from various perspectives. From a learning perspective, the way such models could be estimated from the data, both in the context independent and context dependent cases is still to be analysed;

- (linear) sequential data constitute the simplest form of linguistic data types: sets of multi-stream sequences, finite languages, trees, recursive feature structures... should also be equipped with the same methodological apparatus if one wishes to enlarge the spectrum of applications for analogical learning procedures. This work is also in progress: based on

[6] In [17], Lepage explicitely rules this solution out. It is however unclear why this solution is not valid, and what the consequences of this decision on the rest of his system are.

the work developped in this paper, [29] present a general model of analogical relationship which applies to a large range of algebraic structures, including notably trees and directed acyclic graphs;

- finally, empirical studies on a variety of tasks and languages have to be carried out in order to assess the adequation of our models with various linguistic data.

References

[1] Laurent Blin and Laurent Miclet. Generating synthetic speech prosody with lazy learning in tree structures. In *Proceedings of CoNLL'00*, pages 87–90, Lisboa, Portugal, 2000.

[2] Maxime Crochemore, Christophe Hancart, and Thierry Lecroq. *Algorithmique du texte*. Vuibert, Paris, France, 2001.

[3] Mehdi Dastani, Bipin Indurkhya, and Remko Scha. Analogical projection in pattern perception. *Journal of Experimental and Theoretical Artificial Intelligence (JETAI)*, 15(4):489–511, 2003.

[4] Ferdinand de SAUSSURE. *Cours de linguistique générale*. Payot, Lausanne et Paris, 1995. 1[e] éd. 1916.

[5] Arnaud Delhay and Laurent Miclet. Solving analogical equations for learning by analogy with sequences. In Michel Liquière and Marc Sebban, editors, *Conférence d'Apprentissage 2004 (CAP'2004)*, pages 347–362, Montpellier, France, 2004. PUG.

[6] Brian Falkenhainer, Kenneth D. Forbus, and Dedre Gentner. Structure mapping engine. *Artificial Intelligence*, 41, 1990.

[7] Brian Falkenhainer and Dedre Gentner. The structure-mapping engine. In *Proceedings of the meeting of the American Association for Artificial Intelligence (AAAI)*, pages 272–277, 1986.

[8] Stefano Federici, Vito Pirrelli, and François Yvon. A dynamic approach to paradigm-driven analogy. In *Symbolic, connectionist, and statistical approaches to learning for natural language processing*, Berlin, 1996. Springer-Verlag.

[9] Dedre Gentner, Keith J. Holyoak, and Boicho K. Kokinov. *The analogical mind. Perspectives from Cognitive Science.* The MIT Press, Cambridge, MA, 2001.

[10] Nabil Hathout. Analogies morpho-synonymiques. Une méthode d'acquisition automatique de liens morphologiques à partir d'un dictionnaire de synonymes. In *Actes de TALN 2001*, pages 223–232, Tours, July 2001.

[11] Douglas Hofstadter and the Fluid Analogies Research Group. *Fluid Concepts and Creative Analogies.* Basic Books, New York, 1994.

[12] Keith J. Holyoak and Paul Thagard. Analogical mapping by constraint satisfaction. *Cognitive Science*, 13:295–355, 1989.

[13] Janet Kolodner. *Case-Based Reasoning*. Morgan Kaufmann, San Mateo, CA, 1993.

[14] René-Joseph Lavie. *Le locuteur analogique ou la grammaire mise à sa place.* PhD thesis, Université de Paris 10, 2003.

[15] Yves Lepage. Solving analogies on words: an algorithm. In *Proceedings of COLING-ACL'98*, volume 1, pages 728–735, Montréal, Canada, 1998.

[16] Yves Lepage. Open set experiments with direct analysis by analogy. In *Proceedings of NLPRS-99*, pages 363–368, Beijing, 1999.

[17] Yves Lepage. Défense et illustration de l'analogie. In *Actes de la* $8^{ème}$ *conférence sur le Traitement Automatique des Langues Naturelles (TALN)*, pages 373–377, Tours, France, 2001.

[18] Yves Lepage. De l'analogie rendant compte de la commutation en linguistique. Mémoire d'habilitation à diriger des recherches, 2003.

[19] Yves Lepage and Shin-Ichi Ando. Saussurian analogy: a theoretical account and its application. In *Proceedings of COLING-96*, pages 717–722, København, 1996.

[20] Laurent Miclet and Arnaud Delhay. Analogy on sequences: a definition and an algorithm. Technical Report 4969, INRIA, 2003.

[21] Laurent Miclet and Arnaud Delhay. Relations d'analogie et distance sur un alphabet défini par des traits. Technical Report 5244, INRIA, 2004.

[22] L. Micó, Jose Oncina, and Enrique Vidal. A new version of the nearest-neighbour approximating and eliminating search algorithm (aesa) with linear preprocessing-time and memory requirements. *Pattern Recognition Letters*, 15:9–17, 1994.

[23] Melanie Mitchell. *Analogy-Making as Perception.* MIT Press, Cambridge, MA, 1993.

[24] Tom M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[25] Mehryar Mohri. Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350, 2002.

[26] Vito Pirrelli and François Yvon. Analogy in the lexicon: a probe into analogy-based machine learning of language. In *Proceedings of the 6th International Symposium on Human Communication*, Santiago de Cuba, Cuba, 1999.

[27] Jacques Sakarovitch. *Eléments de théorie des automates.* Vuibert, Paris, France, 2003.

[28] David Sankoff and Joseph B. Kruskal, editors. *Time Warps, String Edits and Macromolecules : the Theory and Practice of Sequence Comparison.* Addidon-Wesley, 1983.

[29] Nicolas Stroppa and François Yvon. Formal models of analogical relationships. Technical report, ENST, Paris, France, Forthcoming, 2004.

[30] Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173, 1974.

[31] François Yvon. Paradigmatic cascades: a linguistically sound model of pronunciation by analogy. In *Proceedings of the 35th annual meeting of the Association for Computational Linguistics (ACL)*, Madrid, Spain, 1997.

[32] François Yvon. Pronouncing unknown words using multi-dimensional analogies. In *Proceeding of the European conference on Speech Application and Technology (Eurospeech)*, volume 1, pages 199–202, Budapest, Hungary, 1999.

[33] François Yvon. Finite-state transducers solving analogies on words. Technical Report D008, Ecole Nationale Supérieure des Télécommunications, Paris, France, 2003.