Finite-state transducers solving analogies on words

François Yvon
GET/ENST & LTCI
46 rue Barrault
F-75013 Paris
yvon@enst.fr

Résumé

Transducteurs à états finis calculant des analogies entre mots.

Ce rapport vise à fournir une définition formelle de la notion de *proportion analogique* entre mots sur un alphabet fini. En partant de cette définition, nous montrons que la résolution d'une équation analogique, c'est-à-dire la détermination du quatrième terme d'une proportion étant donnés les trois autres, peut être effectuée de manière efficace en utilisant des transducteurs à états finis. Nous discutons des implications de ce résultat pour l'apprentissage à base d'analogie, en particulier pour des tâches d'analyse du langage naturel, ainsi que diverses extensions possibles de ce travail.

Abstract

This paper intends to provide a sound definition for the notion of analogical proportion between strings over a finite alphabet. Given this definition, we demonstrate that solving an analogical equation, ie. finding the fourth term of a proportion can be performed using finite-state transducers. Implications for analogy-based learning in the context of Natural Language Processing applications are considered.

1 Introduction

1.1 Analogical Natural Language Processing

Analogy-based learning traditionally refers to a class of so-called *l*azy learning algorithms, which are able to generalize the training instances without actually building an explicit model of the data.

Let us consider a typical (artificial) learning situation [1], where the learner is given a set of n objects $\{(x_i, y_i), i = 1 \dots n\}$, independently drawn from a distribution D.

Each object i is represented by a pair (x_i, y_i), where x_i and y_i denote sets of observable properties, which may take various forms, such as real or discrete valued vectors, strings, trees, recursive feature structures... The learner's goal is to use such data, in order to be able to guess the value of y_i associated with any possible value of x_i. In this situation, a "traditional" learner will assume a functional dependency $y = f(x)$, where f is bound to belong to some functional space $\mathcal{F}$ and induces f from the data by searching the function that minimizes some predefined loss function, such as, for instance, the empirical error rate [2]. f can then be used to process any new input. In comparison, a *lazy learner* [3] simply accumulates the training examples in its memory: on a new input x, the prediction procedure consists in (i) finding a set S of similar known instances in the system's memory: this is the *retrieval step*; (ii) compute a prediction for y, based solely on the elements in S: this is the *transfer step*.

Analogy-based learners differ from other lazy learners in the way they define the similarity between objects: in an analogy-based setting, similarity is primarily based on a *structural mapping* (see e.g. [4, 5]) between instances.

Natural language is fraught with exception and irregularities, and as such is an ideal test bed for lazy learning strategies, which readily incorporate some rote learning abilities. Additional arguments advocating the relevance of analogical approaches for NLP tasks are discussed at length in [6], to which we refer the reader. More to the point, this paper also introduces a general model of multi-dimensional structured analogies. In this model, linguistic objects are represented as a set of feature-value pairs. This bundle of features comprises two subsets: the input feature set, denoted $\mathcal{I}$, and the output feature set, $\mathcal{O}$. Given a new object, for which only $\mathcal{I}$ is known, the goal of the inference procedure is to predict the values for each feature in $\mathcal{O}$. In this context, the retrieval stage extracts all possible *analogical proportions* in the input space; these proportions are used as the basis for the *analogical transfer* of the output features.

For any feature f, an *analogical proportion* relates four values: f_1, f_2, f_3, and f_4, such that "f_1 is to f_2 as f_3 is to f_4", denoted as $f_1 : f_2 = f_3 : f_4$. The proportion can be defined in many different ways, depending on the feature type, and on specific implementation choices. For symbolic feature values, we have f_1, f_2, f_3 and f_4 define an analogical proportion if and only if $(f_1 = f_2$ and $f_3 = f_4)$ or $(f_1 = f_3$ and $f_2 = f_4)$. As for string features, which are not uncommon in NLP applications, the definition of an analogical proportion is less obvious, and various proposals have been made, as for instance in [7, 8, 9, 10, 11]. More will be said shortly on this matter. More complex features, corresponding to structured strings (trees), or recursive feature structures, are rarely considered (see however [12] and [13]). This definition can be generalized to non-recursive sets of features: four objects define a *multi-dimensional* analogical proportion with respect to the feature set $\mathcal{I}$ if and only if they define an analogical proportion over all features in $\mathcal{I}$. Suppose that our objects are lexemes described by their orthographic representation and their part-of-speech, then the following is a valid analogical proportion: *(eat,Verb):(eater,Noun)=(cheat,Verb):(cheater,Noun).* To the contrary, *(imp,Noun):(imply,Verb)=(great,Adj):(greatly,Adv)* does not hold: the proportion only applies to the orthographic level, but it does not in the part-of-speech domain. Early applications of multidimensional analogies are reported in [11].

The *analogical transfer* of a feature f consists in finding the fourth term of an analogical proportion, given the other three. An analogical transfer involves solution of the analogical equation $f_1 : f_2 = f_3 : X$, where X is the unknown value. Again, this solution may vary depending on the feature type. The notion of analogical transfer is generalized to the case of feature sets (or complex objects): given a 3-tuple of known objects, the feature set induced through analogical transfer takes the values resulting from the analogical transfer of each feature, if they are all defined, and is otherwise undefined.

Given these concepts, we can now see how analogies can participate into an inferencial process (see [8, 14]). Inference of $\mathcal{O}$ for a new object T proceeds in two steps:

i) All analogical proportions involving T and three known objects in the input space are retrieved.

ii) For each 3-tuple of known objects extracted in i), the result of the analogical transfer is computed for all the features in the output set. If the analogical transfer succeeds, its result is added to a set of candidate outputs. A final voting procedure selects the most likely candidate, based on various heuristics.

Experiments [6, 11] performed using this model have demonstrated its ability to successfully cope with various linguistic analysis tasks. These experiments have however shown that this model has two major shortcomings: (i) the notion of an analogy between strings is too narrowly defined, thus hurting the overall performance (recall is the main problem here); (ii) from an algorithmic perspective, the computation of all the analogical triples during learning has a cubic time complexity, slowing down the overall process. The contributions of this paper are as follows:

- to provide a clear definition of an analogy between strings using rational operations: shuffle and string complementation;
- to propose an implementation of a finite-state analogy solver;
- outline a possible implementation of a learning procedure;

Possible extensions of this work, notably to rational sets, to probabilistic languages, and to structured strings are finally discussed.

1.2 Notations

In this paper, we will be dealing with words, ie. with finite strings over a finite alphabet Σ. The set of all such strings, including ϵ, the empty string, will be noted $\Sigma^\star$. For x and y in $\Sigma^\star$, we note xy the concatenation of x and y. We also note $|\, x \,|$ the length of x, and $x = x_1 \dots x_{|x|}$ or $x = x_1^{|x|}$, with $x_i \in \Sigma$. $x_j \dots x_k$ and x_j^k denotes the empty string ϵ whenever $k < j$.

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 any word v such that $\exists i_1 \dots i_k, 1 \leq i_1 < i_2 \dots i_k \leq n$ such that $v = w_{i_1} \dots w_{i_k}$. Conversely, any such ordered subset of indices I from $1 \dots n$ induces a subword $w(I)$ of w. The subword relationship will be denoted with the $\in$ symbol. Prefixes, suffixes and factors are special types of subwords, induced by sets of *consecutive* indices. If u is a prefix (resp. suffix) of w, $w = uv$ (resp. $w = vu$) and we note $v = u^{-1}w$ (resp. $v = wu^{-1}$).

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 langage accepted by A is noted $L(A)$; denoting $\delta^\star$ the transitive closure of δ we have $L(A) = \{w, \delta^\star(q^0, w) \in F\}$. Conversely A_L denotes the canonical automaton for L whenever L is a rational language. A finite-state transducer T is a finite 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 $\Sigma_1 = \Sigma_2$. A thorough introduction to finite-state transducers and their use in the context of natural language processing is given in [15].

2 Solving analogies on words

2.1 Analogical proportions

This section introduces the definition of an analogy between words, and discusses some basics properties of our definition.

Definition 1 (Analogical proportions) $(x, y, z, t) \in \Sigma^+$ *define an analogical proportion, noted* $x : y :: z : t$ *if and only if* $\exists n > 0, \alpha_i, i = 1\ldots n, \beta_i, i = 1\ldots n \in \Sigma^\star$ *st. either:*

$$x = \alpha_1\ldots\alpha_n \quad t = \beta_1\ldots\beta_n \quad y = \alpha_1\beta_2\alpha_3\ldots \quad z = \beta_1\alpha_2\beta_3\ldots$$

or

$$x = \alpha_1\ldots\alpha_n \quad t = \beta_1\ldots\beta_n \quad y = \beta_1\alpha_2\beta_3\ldots \quad z = \alpha_1\beta_2\alpha_3\ldots$$

and $\forall i, \alpha_i\beta_i \neq \epsilon$. *The smallest integer n for which this property holds is termed the* degree *of the proportion.*

For instance, we have:

$$receptive : reception :: defective : defection$$

with $n = 3$ and the following fragments: $\alpha_1 = re$, $\alpha_2 = cept$, $\alpha_3 = ion$, $\beta_1 = de$, $\beta_2 = fect$, $\beta_3 = ive$[1]. The degree of this analogical proportion is thus equal to 3. The degree can be seen as a measure of the complexity of a proportion: the smaller the degree, the better the analogy. This corresponds to the intuition that good analogies should preserve large portions of the original objects; the ideal analogy involving identical objects.

Given this definition, the following properties hold (see also [16]):

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

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

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

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

[1]We have favored here the linguistically motivated joint segmentation of these four (linguistic) words, but might as well have chosen $\alpha_1 = r$, $\alpha_2 = ecept$, and accordingly for β_1 and β_2.

[16] denotes properties (3) and (4) respectively under the names of *exchange of the means* and *symetry of equality*.

Additionnally, we can note that the following recursive property holds if $|x| > 1$:

Proposition 1

$$x : y :: z : t \Leftrightarrow \left\{ \begin{array}{l} x_2^{|x|} y :: z_2^{|z|} t \textit{ or} \\ x_2^{|x|} y_2^{|y|} :: z : t \end{array} \right.$$

This property holds by the mere definition of a proportion (1).

2.2 Analogical equations

Solving an analogical equation consists in finding the fourth term of an analogical proportion, given the other three.

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

Not all analogical equations have a solution: for instance, given our definition of a proportion, the equation $abc : def :: ijk :?$ does not have any solution. [16] lists a series of *necessary conditions* for an analogical equation to hold. If t is a solution of $x : y :: z :?$, then we must have:

- Symbol inclusion: noting $\overline{x}$ the set of symbols in $\Sigma^\star$ which do not occur in x, this condition is expressed as:

$$\overline{x} \cup \overline{t} = \overline{y} \cup \overline{z} \tag{5}$$

- Similarity: if $d(x, y)$ is the distance defined over $\Sigma^\star$ as the length of the longest common subsequence in x and y, then the similarity condition is expressed as:

$$\left\{ \begin{array}{lcl} |x| - d(x,y) & = & |z| - d(z,t) \\ |y| - d(y,t) & = & |x| - d(x,z) \\ |z| - d(z,x) & = & |t| - d(t,y) \\ |t| - d(t,z) & = & |y| - d(y,x) \end{array} \right. \tag{6}$$

  which implies that: $|x| + |t| = |y| + |z|$.

In other words, a necessary condition for any kind of proportion involving x, y, z and t to hold is that all symbols in x be found in y or in z. The solution t of a an analogical equation will then contain precisely those symbols in y and z that are not found in x. A consequence is that all the solutions of an analogical equation have the same length.

In the general case, analogical equations can have more than one solution. For instance, $c : ac :: bc :?$, has two equally acceptable solutions: abc and bac.

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 that, given our definition of an analogical proportion between words, this partial function is rational and can be efficiently computed using a finite-state transducer.

2.3 An alternative characterisation

In this section, we show that our notion of an analogical proportion on words can be expressed in the terms of two basic constructions on words and languages, the *shuffle* and the *complementary set* constructions.

2.3.1 Shuffle

The notion of *shuffle* is introduced eg. in [17] as follows. If u and v are two words in $\Sigma^\star$, their shuffle is the language defined as:

$$u \bullet v = \{u_1 v_1 u_2 v_2 \dots u_n v_n,\ \text{with } u_i, v_i \in \Sigma^\star, u = u_1 \dots u_n, w = v_1 \dots v_n\}$$

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, with the additionnal constraint that if a symbol a precedes b in u (or in v), then it must occur before b in w.

Taking for instance $u = abc$ and $v = def$, the following words: $abcdef$, $abdefc$, $adbecf$ ... all belong to $u \bullet v$; this is not the case with $abefcd$, in which d occurs after, instead of before, e.

The shuffle operation has the following basic properties:

$$\begin{aligned} u \bullet \epsilon &= \{u\} \ (\epsilon \text{ is “neutral”}) && (7)\\ u \bullet v &= v \bullet u \ (\text{commutativity}) && (8)\\ (u \bullet v) \bullet w &= u \bullet (v \bullet w) \ (\text{associativity}) && (9)\\ u(v \bullet w) &\subset (uv) \bullet w && (10) \end{aligned}$$

The shuffle is generalized to languages in an intuitive manner according to:

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

As a notational simplification, we will identify $u \bullet v$ and $\{u\} \bullet \{v\}$.

2.3.2 Complementary subwords and complementary sets

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

Definition 3 (Complementary subwords and set) *If x is a subword of w, the complementary set of x with respect to w is defined as $w \backslash x = \{y, y \in w, x = w(\{0 \dots |w|\} \backslash I_y)\}$. If x is not a subword of w, $w \backslash x$ is empty.*

If y is an element in $w \backslash x$, we will say that y is a complementary subword of x in w.

The complementary set of x with respect to w simply contains what “remains” of w when the symbols in x are discarded. For instance, the complementary set of $false$ wrt. $falsehood$ is the singleton $\{hood\}$; the complementary subwords of ive wrt. $derivative$ is the set: $\{deratie, dervati, derivat\}$. This operation can be turned into a (symetric) binary relationship as follows:

Definition 4 (Complementary relationship) *$w \in \Sigma^\star$ define a binary relationship denoted $\backslash_w$ and defined as: $u \backslash_w v$ if and only if $u \in w \backslash v$.*

The complementary set of a word x wrt. a language L generalizes this notion and is defined as:

$$L \backslash x = \bigcup_{w \in L} w \backslash x$$

Similarly, one can also define the complementary set of a language K wrt. a word w as:

$$w \backslash L = \bigcup_{x \in L} w \backslash x$$

This operation is readily extended to languages: if K and L are two languages, the complementary $(\backslash K, L)$ of K with respect to L is the union over words in L of their complementary set with respect to K:

$$L \backslash K = \bigcup_{w \in L, x \in K} w \backslash x = \bigcup_{w \in L} w \backslash K = \bigcup_{x \in K} L \backslash x$$

The notions of complementary sets and shuffle are related through the following property:

Proposition 2

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

We will also need the following property:

Proposition 3

$$\forall u, v \in \Sigma^\star, \forall w \in u : (u \backslash w) \bullet v \subset (u \bullet v) \backslash w$$

Proof:

$$\begin{aligned} x \in (u \backslash w) \bullet v \quad &\Rightarrow \quad \exists y \in u \backslash w, x \in y \bullet v \\ &\Rightarrow \quad \exists y \in u \backslash w, x = y_1^{k_1} v_1^{l_1} \dots y_{k_{n-1}+1}^{k_n} v_{l_{n-1}+1}^{l_n} \end{aligned}$$

As y is a subword of u, each fragment $y_{k_i}^{k_{i+1}}$ is $u(I_i)$, for some index set I_i. For each value of i, we can extend I_i into an interval I'_i, so as to entirely cover u. Take for instance: $I'_1 = [1 \dots max(I_1)], I'_2 = [max(I_1) + 1 \dots max(I_2)], \dots I'_n = [max(I_n) \dots \mid u \mid]$. x', defined as $x' = u(I'_1) v_1^{l_1} \dots u(I'_n)$ is in $u \bullet v$: it contains all the symbols in u and v, and respects the ordering constraints. w being a subword of u, it is a subword of x', which means that $x' \backslash w$ is not empty. Furthermore, we can substract w in such a way that the result is exactly x (see Figure 1). As a consequence, we have $x \in (u \bullet v) \backslash w$.

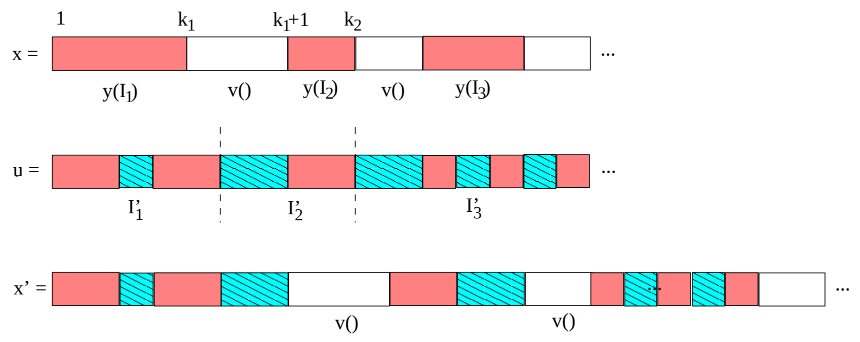

Figure 1: String “substraction”

2.3.3 Analogies revisited

These notions enable us to provide an alternative characterization of the notion of analogy between words, based on the following proposition. This proposition gives a necessary and sufficient condition for an analogical proportion to hold.

Proposition 4

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

Proof. Using the notations of definition (1), we can note that if $x : y :: z : t$, then $\alpha_1\beta_1 \ldots \alpha_n\beta_n$ simultaneously belongs in $x \bullet t$ and to $y \bullet z$, which implies that these two sets have a non-empty intersection. Conversely, let us denote w a word of in $x \bullet t \cap y \bullet z$. If $|w| = 1$, the property trivially holds. Let us assume that it holds up to length n and let us consider a situation where w has length $n+1$: $w = w_1 \ldots w_{n+1}$. w beeing in $x \bullet t$, we must have $w_1 = x_1$ or $w_1 = t_1$; and accordingly for $w_1 = y_1$ or $w_1 = z_1$. Let us assume, without loss of generality, that $w_1 = x_1 = y_1$. Given that w_2^{n+1} belongs both to $x_2^{|x|} \bullet t$ and to $y_2^{|y|} \bullet z$, we can conclude that $x_2^{|x|} : y_2^{|y|} :: z : t$, and then that $x : y :: z : t$.

The following corrollary yields a formal definition of the set of solutions of an analogical equation:

Proposition 5

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

Proof.

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

2.4 Analogical proportions as relations

Analogical proportions enables us to define a binary relationship over $\Sigma^\star \times \Sigma^\star$ as: $(x, y)\mathcal{R}(z, t)$ if and only if $x : y :: z : t$. This relation is clearly reflexive and symetrical, these properties resulting directly from equations (2), (3), and (4).

Given our new characterisation, we can also prove that it is transitive.

Proposition 6 (Transitivity of analogical relationships) *If $x : y :: z : t$ and $z : t :: u : v$, then $x : y :: u : v$*

Proof. Assume that $x : y :: z : t$ and $z : t :: u : v$, and denote w_1 and w_2 two elements respectively of $x \bullet t \cap y \bullet z$ and $z \bullet v \cap t \bullet u$. $w_1 w_2$ belongs to $x \bullet t \bullet z \bullet v$, meaning that $w_1 w_2$ can be spelled out as a sequence of chunks taken from these four words. Discarding the chunks originally in z and t yields a word w which belongs to $x \bullet v$; by the same argument, w also belongs to $y \bullet u$, which implies that these two sets have a non empty intersection, yielding the desired result: $x : y :: z : t$.

3 A Finite-State Analogy Solver

This section details the construction of finite-state transducers for solving analogical equations. Based on the observation that the shuffle and complementary set operation are rational, we detail the construction of a finite-state analogy solver.

3.1 Elementary constructions

In this section, we introduce the elementary finite-state automata computing the subwords of a rational language, the complementary set of a rational language and the shuffle of two languages.

subword

Proposition 7 (Subword is rational) *If L is a rational language, the set of subwords of L, defined as $Sub(L) = \{x, \exists w \in L, x \in w\}$ is also rational.*

The following proof directly introduces the automata recognizing the set of subwords of a rational language L. From $A = (\Sigma, Q, q^0, F, \delta)$, the canonical automaton for L, derive $A' = (\Sigma, Q, q^0, F, \delta')$, where $\delta'(q, x) = r$ if and only if $\delta(q, x) = r$, and $\delta'(q, \epsilon) = r$ if and only if $\exists x \in \Sigma, \delta(q, x) = r$. In other words, each transition in A appears twice in A': once with the original label, and once with the original label turned into an ϵ.

$\forall x \in Sub(L), \exists w \in L$ st. $x \in w$; if $q_1 \dots q_n$ denotes the sequence of states accepting w in A, then $q_1 \dots q_n$ accepts x in A'. Conversely, if $x \in L(A')$, x is accepted along a path $q_1 \dots q_n$ in A'; the same computation also exists in A and spells out a word w such that $x \in w$.

The construction of this automata is illustrated by Figure 2.

The subwords of L can also be computed as a transduction of L through a machine which non-derminislically erases or copies the input symbols (see Figure 2.(b)).

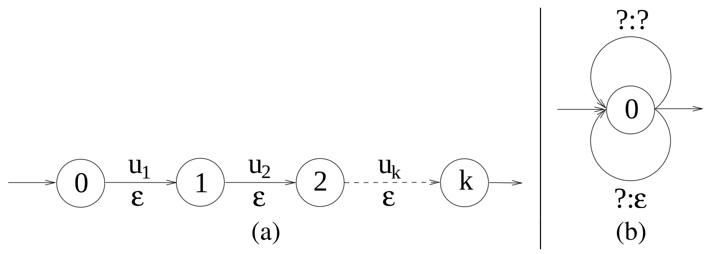

Figure 2: The automata (a) and transducer (b) computing the subwords of u. The placeholder '?' stands for any symbol in Σ.

Complementary set

Proposition 8 *If K and L are rational sets, $L\backslash K$ is also a rational set.*

Proof. We first show that if L is rational, then $L\backslash x$ is rational for any x. From $A_L = (\Sigma, Q, q^0, F, \delta)$, the canonical automaton for L, we derive the finite state transducer $T_L = (\Sigma, Q, q^0, F, \delta')$, where δ' is defined as:

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

It is routine to verify that $T_L(x) = L\backslash x$. If $y \in T_L(x)$, $\exists \sigma = q_1 \dots q_n \in T_L$ such that (i) $q_1 = q^0$, (ii) $q_n \in F$ (iii) the catenation of the input (resp. output) labels along σ is x (resp. y). σ also exists in A_L and spells out a unique word $w \in L$; given the definition of δ', both x and y are subwords of w and stand in a complementary relationship with respect to w. This yields that $y \in L\backslash x$.

Conversely, if y is in $L\backslash x$, there exists a word w in L and an index set I such that $x = w(I)$ and $y = w(\{0 \dots |w|\}\backslash I)$. Consider the following traversal of T_L: starting from $q = q^0$ and $l = 1$, choose $\delta(q, w_l : \epsilon)$ whenever $l \in I$, $\delta(q, w_l : \epsilon)$ otherwise, and increment l; stop for $l = n$. We first observe that such a choice is always possible: it is possible for $l = 0$; if after l steps we reach q, then $q = \delta^\star(q^0, w_1^l)$, meaning that q must have in T_K one outgoing transition labeled $\epsilon : w_{l+1}$ and one labeled $w_{l+1} : \epsilon$. This path reaches a final state in T_L (because $w \in L$), and collects on its output tape all the w_i such that $i \notin I$: this is precisely the definition of y. It follows that $L\backslash K$, being the union over a rational set (K) of the application of a rational transduction, is also a rational set, computed by the machine $T_L \circ A_K$.

Shuffle As noted, for instance in [17], 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 $q_K = r_K$.

The shuffle $u \bullet v$ is also computed as the image of u (resp. v) through the machine which non-deterministically either copies its input or inserts a symbol in v (resp. v).

3.2 Solving Analogies on words

With these basic constructions in place, our 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 built according to the constructions hereabove detailed. The following proposition thus comes as a direct corrolary of proposition (5).

Proposition 9 *The solutions of the analogy $x : y :: z :?$ define a rational set; this set is computed as the $M_{y,z}(x)$.*

3.3 Computing the analogical extension of a language

The problem of finding the analogical extension $\mathcal{A}(L)$ of a language L corresponds to the computation of the set of all words x such that there exists some triple (y, z, t) in L^3 such that $x : y :: z : t$. This computation is a necessary step to performing analogical inference (see Section 1.1).

When L is rational, L can be represented by a canonical deterministic FSA A_L. The computation of $\mathcal{A}(L)$ can be readily performed as follows:

- compute $L \bullet L$ using the shuffle product construction
- compute $M_{L,L} = T_{L\bullet L}$. By definition of the shuffle operation, $L \bullet L$ contains all words x such that $\exists (y, z) \in L \times L, x \in y \bullet z$; this implies that the rational relation computed by $M_{L,L}$ contains all the pairs (x, t) such that $\exists y, z \in L \times L, \exists w \in y \bullet z, t \in w\backslash x$.
- finally compose $M_{L,L}$ with A_L, to get the desired result[2]. Note that the non-determistic machine for L can be quite gigantic: a naive implementation would require $O(n^3)$ states when A_L has n states. However, this machine can be expanded on-the-fly and needs not be explicitely constructed in memory.

Note that we have $L \subset \mathcal{A}(L)$, as for every word $w \in L$: the trivial analogy $w : w :: w : w$ holds.

This 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.

3.4 Computing the degree of an analogical proportion

The degree (see Definition 1) is a possible measure of the quality of an analogy, corresponding to the number of discontinuities involved in a given proportion. Computing this value can be done using a weighted finite-state transducer as follows. Let us consider the analogical equation $x : y :: z :?$, and let g_1 and g_2 be two mappings from Σ to respectively Σ_1 and Σ_2, with $\Sigma_1 \cap \Sigma_2 = \emptyset$. We will also need two fresh symbols $\#_1$ and $\#_2$. Let us now modify our construction of the machine $M_{y,z}$ as follows: instead

[2]As pointed out by N. Stroppa (personal communication), we can automatically keep track of the computation by labelling arcs with 4-uples instead of pairs, and directly output the triple of words involved in the analogy.

of $y \bullet z$, we consider $g_1(y) \bullet g_2(z)$; the complementary set construction is also amended: transitions $\delta(q, a : \epsilon)$ are replaced with $\delta(q, a : \#_1)$ if $a \in \Sigma_1$ (resp. $\delta(q, a : \#_2)$ if $a \in \Sigma_2$). $M_{y,z}$ thus maps words from Σ^* to $(\Sigma_1 \cup \Sigma_2 \cup \{\#_1, \#_2\})^*$. These modifications enable us to trace in the output $M_{y,z}(x)$ the portions which were originally in y (the output symbols are in Σ_1) or in z (the output symbols are in Σ_2).

Let $T_d = (\Sigma_1 \cup \Sigma_2, Q, q_0, \delta)$ be a 5-states weighted transducer defined as: $Q = \{0, 1, 2, 3, 4\}$, $I = 0$, $F = \{1, 2, 3, 4\}$,and δ is defined as:

- initial arcs: $\forall a \in \Sigma_1, \delta(0, a : g_1^{-1}(a)/1) = 1$; $\delta(0, \#_1 : \epsilon/1) = 2$; $\forall a \in \Sigma_2, \delta(0, a : g_2^{-1}(a)/1) = 3$; $\delta(0, \#_2 : \epsilon/1) = 4$.
- loops: $\forall a \in \Sigma_1, \delta(1, a : g_1^{-1}(a)/0) = 1$; $\delta(2, \#_1 : \epsilon/0) = 2$; $\delta(3, \#_2 : \epsilon/0) = 3$; $\forall a \in \Sigma_2, \delta(4, a : g_2^{-1}(a)/0) = 3$;
- other transitions:
  - $\delta(1, \#_1 : \epsilon/1) = 2$; $\delta(1, \#_2 : \epsilon/0) = 3$; $\forall a \in \Sigma_2, \delta(1, a : g_2^{-1}(a)/1) = 4$;
  - $\forall a \in \Sigma_1, \delta(2, a : g_1^{-1}(a)/1) = 1$; $\delta(2, \#_2 : \epsilon/1) = 3$; $\forall a \in \Sigma_2, \delta(2, a : g_2^{-1}(a)/0) = 4$;
  - $\forall a \in \Sigma_1, \delta(4, a : g_1^{-1}(a)/1) = 1$; $\delta(4, \#_1 : \epsilon/0) = 2$; $\delta(4, \#_2 : \epsilon/1) = 4$;
  - $\forall a \in \Sigma_1, \delta(3, a : g_1^{-1}(a)/0) = 1$; $\delta(3, \#_1 : \epsilon/1) = 2$; $\forall a \in \Sigma_2, \delta(3, a : g_2^{-1}(a)/1) = 4$;

This finite-state machine is displayed on Figure 3.

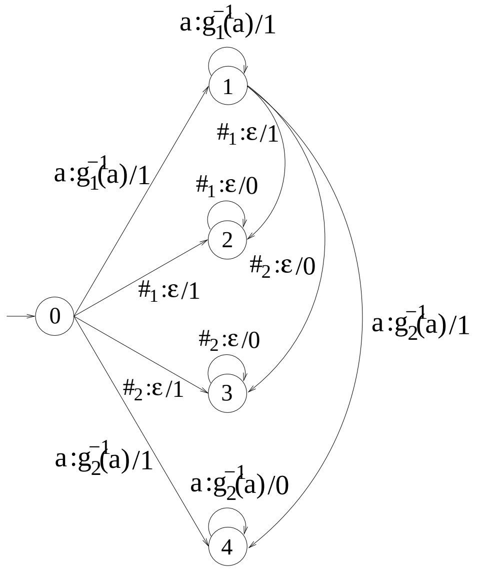

Figure 3: The weighted FST T_d computing the degree of an analogy.
Only the transitions for states 0 and 1 are fully represented. States 1 to 4 are final.

$C_{y,z}$ is defined as the composition of letter-to-letter transducers: $M_{y,z} \circ T_d$ and outputs pairs $(t, d(t))$ in $\Sigma \times N$. The degree of an analogical proportion is computed as $d(x : y :: z : t) = min\{d(t), (t, d(t)) \in C_{y,z}(x)\}$.

Proposition 10 *The degree of* $x : y :: z : t$ *is computed as:* $min\{d(t), (t, d(t)) \in C_{y,z}(x)\}$

Proof. Suppose $(t, d(t)$ is in $C_{y,z}(x)$, with t' the corresponding output of $M_{y,z}$. t' is a word over $\Sigma_1 \cup \Sigma_2 \cup \{\#_1, \#_2\}$, and can be unambiguously decomposed in maximally long homogeneous chunks ${t'}_i, i = 1 \dots k$, such that each t'_i is either in Σ_1^*, Σ_2^*, $\{\#_1\}^*$ or $\{\#_2\}^*$. We readily derive a joint segmentation of x, y, z, t in $d(t)$ chunks based on the following procedure (starting with $i = 1, j = 1$):

- if $t'_i \in \Sigma_1^*$, then $t_i = y_i = g_1^{-1}({t'}_i)$, $x_i = z_i = \epsilon$
- if $t'_i \in \Sigma_2^*$, then $t_i = z_i = g_1^{-1}({t'}_i)$, $x_i = y_i = \epsilon$
- if $t'_i = \#_1^m$, then $t_i = z_i = \epsilon$, $x_i = y_i = x_j^{j+m+1}$; j is then updated according to: $j = j + m + 1$
- if $t'_i = \#_2^m$, then $t_i = y_i = \epsilon$, $x_i = z_i = x_j^{j+m+1}$; j is then updated according to: $j = j + m + 1$

Note further that adjacent chunks corresponding to null cost transitions in T_d can be merged: for instance, if we have $t'_i \in \Sigma_1^*$ and $t'_{i+1} = \#_2^m$ for some m, we can simulaneously merge $t_i = y_i = g_1^{-1}({t'}_i)$ with $t_{i+1} = y_{i+1} = \epsilon$ and $x_i = z_i = \epsilon$ with $x_{i+1} = z_{i+1} = x_j^{j+m+1}$. This process is iterated until all possible merges have been performed, leaving us with exactly $d(t)$ chunks satisfying Definition 1.

Conversely, let us assume $x : y :: z : t$; then $\exists \alpha_i, i = 1 \dots n, \beta_i, i = 1 \dots n \in \Sigma^\star$ st. $x = \alpha_1 \dots \alpha_n, t = \beta_1 \dots \beta_n, y = \alpha_1 \beta_2 \alpha_3 \dots, z = \beta_1 \alpha_2 \beta_3 \dots$ and $\forall i, \alpha_i \beta_i \neq \epsilon$, and n is the smallest integer for which this property holds.

A transduction of x through $M_{y,z}$ is readily derived using the following procedure:

- if i is odd: read α_i on the input tape and output $t'_{2i-1} = \#_1^{|\alpha_i|}$: such a path exists in $T_{\{y\}}$ and remains in $M_{y,z}$; then follow a sequence of epsilon transitions and output $t'_{2i} = g_2(\beta_i)$: this path exists in $T_{\{z\}}$ and remains in $M_{y,z}$.
- if i is even: read α_i on the input tape and output $t'_{2i-1} = \#_2^{|\alpha_i|}$: such a path exists in $T_{\{z\}}$ and remains in $M_{y,z}$; then follow a sequence of epsilon transitions and output $t'_{2i} = g_1(\beta_i)$: this path exists in $T_{\{z\}}$ and remains in $M_{y,z}$.

This procedure ensures that at least one element t' of $M_{y,z}(x)$ is such that $T_d(t') = n$. This is because:

- the initial transition has a unit cost
- transitions from t'_{2i-1} to t'_{2i} incur a null cost;
- transitions from t'_{2i} to t'_{2i+1} incur a unit cost.

Altogether we have shown that at least one transduction $(M_{y,z} \circ T_d)(x)$ finally outputs $(t, d(x : y :: z : t))$.

4 Related work

The modeling of analogical proportions between strings has so far received little attention. Most relevant however to our discussion is the work of Yves Lepage, especially [10] and [16], presented in full details in [18].

[10] details the implementation of a first analogy solver for words, based on a generalization of algorithms computing the edit distance between strings. Lepage first introduces a dissimility measure $pdist(u, v)$ between strings u and v. In essence, $pdist(u, v)$ is computed as the traditional edit distance [19], using however 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.

It is a well know fact that edition distances can be computed using weighted finite-state transducers (see eg. [20]). For instance, denoting C_u the copy transducer for word u, the dissimilarity $pdist(u, v)$ is readily computed as a shortest path in $C_u \circ T_p \circ C_v$, where T_p is represented on Figure 4.

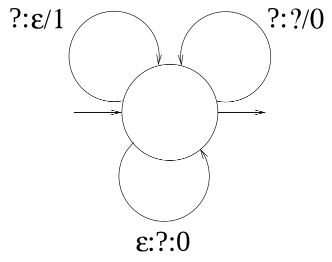

Figure 4: Transducer computing the (pseudo)-edit distance
The placeholder ? stands for any symbol.

Step (1) of the previous algorithm can then be restated as finding a minimal cost path for the transduction of C_x in $T_y = (T_p \circ C_z)$ and $T_z = (T_p \circ C_y)$. Note that the machine $T_y \cap T_z$ has the same set of states as $M_{y,z}$: one for each pair (q, r), where q is a state of C_x and r a state of C_z. Step (2) of the algorithm simulates a traversal of x in $T_y \cap T_z$, using heuristics to select alternative choices: at each iteration of step (2) 9 choices are considered, which correspond to 9 possible moves in the intersection automaton. Examining these alternatives and the corresponding output symbols, it appears that the same computation exists in our analogical transducer $M_{y,z}$. We therefore claim that the solutions produced using this analogical solver are always found in our own model, which has the benefits of making the inherent non-determinism of analogical equations explicit: where Lepage's model resorts to heuristics to select one possible solution, our finite-state machine computes the set of all possible solutions, each with the corresponding degree. Analogical solvers based on edit distances have recently been reinvestingated in [21].

[16] goes one step further, and proposes a more powerful model of analogy between words, capable of modeling non-rational languages. Let us review here briefly his main arguments. 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 that we have derived here as direct consequences on our own definition. Lepage then considers the effect of the concatenation operation on analogical proportions 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 $x_1 : y_1 \overset{\mathrm{G}}{=} z_1 : t_1$ and $x_2 : y_2 \overset{\mathrm{G}}{=} z_2 : t_2$, 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, he introduces the following generative model: Given M, a finite set of pairs from $\Sigma^\star$, $x \vdash_M y$ if $\exists (z,t) \in M$ such that $x : y \overset{\mathrm{PS}}{=} z : t$. Denoting $\vdash_M^\star$ the transitive closure of this relation, he finally defines the langage of analogical strings generated from A, some finite set 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: $L_M(A) = \{a^n b^n\}$. To prove this result, Lepage introduces two new sets of analogical patterns, presented as *true analogies* and supposed to hold for every value of n: $a : aa \overset{\mathrm{G}}{=} a^{n-1} : a^n$ and $b : bb \overset{\mathrm{G}}{=} b^{n-1} : b^n$. The concatenation axiom (axiom 1) thus allows to derive the following analogy: $a^{n-1}b^{n-1} : a^n b^n \overset{\mathrm{G}}{=} ab : aabb$, 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 M. Note that another concatenation rule could also apply: $b^{n-1}a^{n-1} : b^n a^n \overset{\mathrm{G}}{=} ab : aabb$: 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 $a^n b^n$. But why restrict to these kinds of "compound" analogies: in fact, solving $aabb$:? $\overset{\mathrm{G}}{=} ab : aabb$, we find that $aababb$, which does not belong to $\{a^n b^n\}$, can also be derived[3]from ab. From there, we can derive many more words not in $\{a^n b^n\}$. The additional restriction to use solely "compound" analogies, *based on a set of primitive patterns holding independantly for disjoint subsets of the alphabet*, indeed makes the model more constrained

[3]In [22], 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.

and allows to generate non rational languages[4]. This restriction however requires a more complex model than the one we (and to a large extent Lepage himself) used in the rest of our work.

5 Conclusion and future work

In this paper, we have developped an algebraic framework for modeling the notion of analogy between strings over finite sets of symbols. We have also shown how our definition of analogy could be efficiently implemented using weighted finite-state transducers. What remains to be done is (i) integrate this model of analogy in an inferencial process, and (ii) study the empirical potential of analogical inference for various NLP tasks.

This, however, is only the first step of a long-term research program, and many promising extensions of this work need be considered. This notably includes:

- generalize the notion of an analogical proportion to the case of finite languages, then to rational languages. The first case is especially significant in the context of NLP applications, as many linguistic analysis tasks can be defined as a mapping between a word with a finite language. This is exemplarly the case when some non-determinism is involved, as is the case with the pronunciation task : rather than viewing this task as mapping between an orthographic word and a phonemic representation of its pronunciation, it should be better modeled as a mapping between orthography and a (finite) set of equally likely pronunciations. Many tasks in morphology can be cast in the same framework: conjugation, for instance, is a mapping of a lexeme into a finite set of surface forms, one for each slot in the morphological paradigm.

  Concatenation being an internal operation for finite and rational languages. definition 1 can readily be extended to languages. However, how the various algorithms for solving analogical equations can also be generalized remains to be seen.

- generalize the notion of an analogical proportion to weighted (eg. probabilistic) languages. We have, thus far, considered the quality of an analogy to be directly related to its degree. However, other criteria may also come into play: for instance, depending on the languages, analogical proportions corresponding between morphologically related terms may "occur" closer to one end of a word; the frequency of the terms involved in a proportion may also play a significant role [5] . This suggests that it might prove useful, especially in the context of analogical inference, to look into the properties of analogical proportions in weighted or probabilistic languages.

[4]Our understanding of the joint use of (i) a finite set of primitive words, (ii) a finite number of "analogical pairs" (iii) a finite number of "true analogies" allows to *constraint rewriting operations to apply in parallel*, a property which makes Lepage's system reminiscent of L systems.

[5]It is for instance an admitted fact that sub-families of irregular verbs are less prone to undue generalization if they include frequently occurring verbs [23].

- explore the property of analogical proportions involving more (or less) than 4 terms. In fact, definition 1 can be generalised in many ways. One could for instance extend analogical proportions to involve more (or less !) than 4 terms. One would, for instance, define 5-ways analogies as $x : y : z :: u : v$ if and only if $x \bullet y \bullet z \cap u \bullet v \bullet \neq \emptyset$, or 6-ways analogies to $x : y : z :: u : v : w$ if and only if $x \bullet y \bullet z \cap u \bullet v \bullet w \neq \emptyset$.

  An examplar 6-way analogy in the domain of orthographic strings is the following:

  $$poison : foisonner : empeser :: foison : empoisonner : peser$$

  Such generalizations might prove useful in contexts where 4-ways proportions are seldom found, thus limitating the power of the analogical inference process.

  Again, the implications of such extensions are to be carefully considered, especially in terms of their formal complexity[6].

- study how additional linguistic constraints could be included in the model. For instance, one can imagine a restriction of the subword relationship which would rule out careful chosen transitions. Assume, for instance, that words y and z can only be split at certain pre-defined split-points[7]: then we could prohibit the computation of subwords crossing these boundaries, thus making the analogical process more constraint.

- extend the notion of analogy to more complex representations such as trees or graphs. In spite of the centrality of these representations in the linguistic tradition, very few attempts have been made in that direction (see however [12] and [25]). A first step in providing a sound algebraic framework for modeling analogy in such domains will be to study direct extensions of our model to trees, using substitution as the primary operation between (labelled) trees.

Acknowledgements

The author wishes to thank Vito Pirrelli, who has accompanied and stimulated much of the work reported here. Stefano Federici, Yves Lepage, Laurent Miclet, Arnaud Delhaye, Antoine Cornuéjols and Nicolas Stroppa have provided very significant to this work, and should be rightly rewarded for their contribution.

References

[1] T. M. Mitchell, *Machine Learning*. McGraw-Hill, 1997.

[2] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Springer, 1995.

[6]The complexity of the n-way shuffle is for instance studied in [24].

[7]For instance, in a linguistic application, these could be syllable boundaries in the case of linguistic words, or chunk boundaries in a syntactic application

[3] W. Daelemans, “Abstraction considered harmful: lazy learning of language processing,” in *Proceedings of the sixth Belgian-Dutch Conference on Machine Learning* (H. J. V. den Herik and A. Weijters, eds.), (Maastricht, The Nederlands), pp. 3–12, 1996.

[4] D. Gentner, “The mechanisms of analogical learning,” in *Similarity and Analogical Reasoning* (V. S. and O. A., eds.), pp. 199–241, Cambridge University Press, 1989.

[5] B. Falkenhainer, K. D. Forbus, and D. Gentner, “Structure mapping engine: algorithm and example,” *Artificial Intelligence*, vol. 41, pp. 1–63, 1990.

[6] V. Pirrelli and F. 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.

[7] M. Mitchell and D. R. Hofstadter, “The emergence of understanding in a computer model of concepts and analogy-making,” *Physica*, vol. 42, pp. 322–334, 1990.

[8] S. Federici, V. Pirrelli, and F. Yvon, “A dynamic approach to paradigm-driven analogy,” in *Symbolic, connectionist, and statistical approaches to learning for natural language processing*, (Berlin), Springer-Verlag, 1996.

[9] F. 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.

[10] Y. Lepage, “Solving analogies on words: An algorithm,” in *Proceedings of COLING-ACL’98*, vol. 2, (Montréal, Canada), pp. 728–735, 1998.

[11] F. Yvon, “Pronouncing unknown words using multi-dimensional analogies,” in *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*, vol. 1, (Budapest, Hungary), pp. 199–202, 1999.

[12] Y. Lepage, “Open set experiments with direct analysis by analogy,” in *Proceedings of NLPRS ’99*, vol. 2, (Beijing, China), pp. 363–368, 1999.

[13] L. Blin and L. Miclet, “Generating synthetic speech prosody with lazy learning in tree structures,” in *Proceedings of CoNLL’00*, (Lisboa, PT), pp. 87–90, 2000.

[14] V. Pirrelli and F. Yvon, “The hidden dimension: paradigmatic approaches to data-driven natural language processing,” *Journal of Experimental and Theoretical Artificial Intelligence (JETAI), Special Issue on Memory-Based Language Processing*, vol. 11, pp. 391–408, 1999.

[15] E. Roche and Y. Schabes, “Introduction to finite-state devices in natural language processing,” in *Finite State Natural Language Processing* (E. Roche and Y. Schabes, eds.), (Cambridge, MA), The MIT Press, 1997.

[16] Y. Lepage, “Défense et illustration de l’analogie,” in *Actes de la 8eme conférence sur le Traitement Automatique des Langues Naturelles (TALN 2001)*, (Tours, France), pp. 373–377, 2001.

[17] J. Sakarovitch, *Eléments de théorie des automates*. Vuibert, Paris, 2003.

[18] Y. Lepage, “De l’analogie rendant compte de la commutation en linguistique.” Mémoire d’habilitation à diriger des recherches. Université de Grenoble, 2003.

[19] R. A. Wagner and M. J. Fischer, “The string-to-string correction problem,” *Journal of the ACM (JACM)*, vol. 21, no. 1, pp. 168–173, 1974.

[20] M. Mohri, F. Pereira, and M. Riley, “The design principles of a weighted finite-state transducer library,” *Theoretical Computer Science*, no. 231, pp. 17–32, 2000.

[21] L. Miclet and A. Delhay, “Analogy on sequences: a definition and an algorithm,” in *Actes de la Conférence sur l’Apprentissage (CAP 2003)*, (Laval), 2003.

[22] Y. Lepage, “Analogy and formal language,” *Electronic notes in Theoretical Computer Science*, vol. 47, pp. 1–12, 2001.

[23] S. Pinker and A. Prince, “On language and connectionism: analysis of a parallel distribution processing model of language acquisition,” *Cognition*, vol. 28, pp. 73–194, 1988.

[24] C. Allauzen, “Calcul efficace du shuffle de k mots,” Tech. Rep. 2000-02, Institut Gaspard-Monge, Université de Marne-la-Vallée, 2000.

[25] L. Blin and M. Edgington, “Prosody prediction using a tree-structure similarity metric,” in *Proceedings of the International Conference on Spoken Langage Processing (ICSLP)*, vol. III, (Beijing, China), pp. 183–186, 2000.