Issues in Analogical Inference Over Sequences of Symbols: A Case Study on Proper Name Transliteration

Philippe Langlais and François Yvon

Abstract Formal analogies, that is, proportional analogies involving relations at a formal level (e.g. *cordially* is to *cordial* as *appreciatively* is to *appreciative*) have a long history in Linguistics. They can accommodate a wide variety of linguistic data without resorting to *ad hoc* representations and are inherently good at capturing long range dependencies between data. Unfortunately, applying analogical learning on top of formal analogy to current Natural Language Processing (NLP) tasks, which often involve massive amount of data, is quite challenging. In this chapter, we draw on our previous works and identify some issues that remain to be addressed for formal analogy to stand by itself in the landscape of NLP. As a case study, we monitor our current implementation of analogical learning on a task of transliterating English proper names into Chinese.

1 Introduction

A proportional analogy is a relationship between four objects, denoted $[x : y :: z : t]$, which reads as "x is to y as z is to t". While some studies have been devoted to handle semantic relationships [1, 2], we focus in this work on *formal* proportional analogies (hereafter formal analogies or simply analogies), that is, proportional analogies involving relationships at the formal level, such as [*miracle* : *miraculous* :: *fable* : *fabulous*]. This is different from analogies considered by computational models of analogy-making such as Copycat [3] and Metacat [4, 5]. Those systems do learn to solve puzzles on letters such as [*abc* : *abd* :: *mrrjj* : ?]. In these studies, letters are

P. Langlais (✉)
Université de Montreal, Montreal, Canada
e-mail: felipe@iro.umontreal.ca

F. Yvon
LIMSI/CNRS, Université Paris Sud, Orsay, France
e-mail: yvon@limsi.fr

H. Prade and G. Richard (eds.), *Computational Approaches to Analogical Reasoning: Current Trends*, Studies in Computational Intelligence 548,
DOI: 10.1007/978-3-642-54516-0_3, © Springer-Verlag Berlin Heidelberg 2014

nothing more than abstractions with a few meaningful relations (i.e., successorship, predecessorship and sameness) with which the system reasons.

Early work on formal analogies for Natural Language Processing (NLP) was devoted to propose computational definitions of proportional analogies. Yvon [6] proposed a definition where a prefixation or a suffixation operation was allowed between forms. In [7], Lepage proposed a richer model allowing at the same time, prefixation, suffixation, as well as infixation operations. His model is characterized in terms of the edit-distance that must verify four entities in (formal) analogical relation. Later on, Yvon et al. [8, 9] proposed a model of analogy which generalizes the model of [7] thanks to finite-state machines. In particular, this model can account for inversions (i.e. *Paul gave an apple to Mary* is to *Mary received an apple from Paul* as *Paul gave a letter to Mary* is to *Mary received a letter from Paul*). Stroppa and Yvon [10] further extended this model to various algebraic structures, notably trees, which are ubiquitous in NLP. Also, Miclet et al. [11] built on the definition of [9] and defined the notion of *analogical dissimilarity* on forms. Presumably, allowing near analogies might be of interest in several AI applications. An extension of analogical dissimilarity to tree structures has been recently proposed in [12].

Another thread of studies is devoted to applications of analogical learning to NLP tasks. Lepage [7] early proposed an analogical model of parsing which uses a treebank (a database of syntactically analyzed sentences). He conducted proof-of-concept experiments. Yvon [6] addressed the task of grapheme-to-phoneme conversion, a problem which continues to be studied thoroughly (e.g. [13]). In [14], the authors address the task of identifying morphologically related word forms in a lexicon, the main task of the MorphoChallenge evaluation campaign [15]. Their approach, which capitalizes on formal analogy to learn relations between words proved to be competitive with state-of-the-art approaches (e.g. [16]) and ranked first on the Finnish language according the EMMA metric (see [17]) which is now the official metric since Morphochallenge 2010. Stroppa and Yvon [18] applied analogical learning to the computation of morphosyntactic properties associated with a word form (lemma, part-of-speech, and additional features such as number, gender, case, tense, mood, etc.). The performance of the analogical learner on the Dutch language was as good as or better than the one reported in [19].

Lepage and Denoual [20] pioneered the application of analogical learning to Machine Translation. Different variants of the system they proposed have been tested in a number of evaluation campaigns (see for instance [21]). Langlais and Patry [22] investigated the more specific task of translating unknown words, a problem simultaneously studied in [23]. In [24], the authors applied analogical learning to translating terms of the medical domain in different language directions, including some that do not share the same scripts (e.g. Russian/English). The precision of the analogical engine was higher than the one of a state-of-the-art phrase-based statistical engine [25] trained at the character level, but the recall was lower. A simple combination of both systems outperformed significantly both engines. See [26] for a technical discussion of those works. Very recently, Gosme et Lepage [27] studied the use of formal analogy for smoothing n-gram language models.

Analogical learning has also been applied to various other purposes, among which terminology management [28], query expansion in Information Retrieval [29], classification of nominal and binary data, handwritten character recognition [11], as well as for solving Raven IQ tests [30]. All these studies witness that analogical learning based on formal analogies can lead to state-of-the-art performance in a number of applications. Still, it encompasses a number of issues that seriously hinder its widespread use in NLP [26]. This motivates the present chapter.

In the remainder of this chapter, we first describe in Sect. 2 the principles behind analogical learning. Then, we provide in Sect. 3 an account of the issues involved in deploying analogical inference over strings in typical NLP tasks. Section 4 reports our experiments in applying analogical learning on the task of transliterating English proper names into Chinese. In Sect. 5, we conclude our work and identify a number of avenues that deserve further investigations.

2 Principles

In order to understand the methodology, we first clarify the process of analogical learning. Let $\mathcal{L} = \{(i(x_k), o(x_k))_k\}$ be a training set gathering pairs of input $i(x_k)$ and output $o(x_k)$ representations for instances x_k. We call *input set*, denoted $\mathcal{I} = \bigcup_k i(x_k)$, the set of input-space representations in the training set. Given an element t for which only $i(t)$ (or alternatively $o(t)$) is known, analogical learning works by:

1. building $\mathcal{E}_i(t) = \{(x, y, z) \in \mathcal{L}^3 \mid [i(x) : i(y) :: i(z) : i(t)]\}$, the set of triplets in the training set that stand in analogical proportion with t in the input space,
2. building $\mathcal{E}_o(t) = \{u \mid [o(x) : o(y) :: o(z) : u]$ and $(x, y, z) \in \mathcal{E}_i(t)\}$, the set of solutions to the *analogical equations* obtained in the output space,
3. aggregating the solutions in $\mathcal{E}_o(t)$ in order to select $o(t)$.

In this description, $[x : y :: z : t]$ is our notation for a (formal) proportional analogy[1]; and $[x : y :: z : ?]$ is called an analogical equation and represents the set of it solutions. In the sequel, we call x-form, y-form, z-form and t-form the first, second, third and fourth forms respectively of $[x : y :: z : t]$. Also, we sometime refer the two first steps of the inference procedure as the *generator*, while we call the third one the *aggregator*.

Let us illustrate this on a tiny example where the task is to associate a sequence of part-of-speech (POS) tags to any given sentence, viewed as a sequence of words. Let $\mathcal{L} = \{$(*he loves her*, PRP VBZ PRP), (*she loved him*, PRP VBD PRP), (*he smiles at her*, PRP VBZ IN PRP)$\}$ be our training set which maps sequences of words (input) to sequences of POS tags (output). Tagging a (new) sentence such as *she smiled at him*, involves: (i) identifying analogies in the input space: [*he loves her* : *she loved him* :: *he smiles at her* : *she smiled at him*] would be found, (ii) solving the corresponding

[1] We also use $[x : y :: z : t]$ as a predicate.

equations in the output space:[PRP VBZ PRP : PRP VBD PRP :: PRP VBZ IN PRP : ?] would be solved, and (iii) selecting the solution. Here, PRP VBD IN PRP would be the only solution produced.

There are three important aspects to consider when deploying the above learning procedure. First, the search stage (step-1) has a time complexity which is prohibitive in most applications of interest (cubic in the size of $\mathcal{I}$). Second, the aggregation (step-3) of the possibly numerous spurious[2] solutions produced during step-2 is difficult. Last, it might happen that the overall approach does not produce any solution at all, simply because no input analogy is identified during step-1, or because the input analogies identified do not lead to analogies in the output space (failure of the inductive bias).

3 Issues with Analogical Learning

From now on, we focus our discussion to the case of formal analogies over strings. Section 5 expands the discussion to (formal) analogies over other structures, such as trees.

3.1 Formal Analogy

We mentioned that several definitions of formal analogy have been proposed. Two of them seem to stand above the others, in the sense that they can account for a larger variety of relations than the others: the proposal of [7] and the one introduced in [8], then generalized in [9]. The latter, which encompasses the former, is defined in terms of *d-factorization*. A d-factorization of a string x over an alphabet Σ, noted f_x, is a sequence of d *factors* $f_x \equiv (f_x^1, \ldots, f_x^d)$, where $f_x^i \in \Sigma^*$ for all i, and such that $f_x^1 \odot f_x^2 \odot f_x^d \equiv x$; $\odot$ denotes the concatenation operator. In [9], the authors define an analogy as follows.

Definition: $\forall x, y, z$ and $t \in \Sigma^\star$, $[x : y :: z : t]$ **iff** there exists a 4-uple of d-factorizations (f_x, f_y, f_z, f_t) of x, y, z and t respectively, such that $\forall i \in [1, d]$, $(f_y^i, f_z^i) \in \left\{(f_x^i, f_t^i), (f_t^i, f_x^i)\right\}$. The smallest d for which this definition holds is called the *degree* of the analogy.

For instance, [*This guy drinks too much* : *This boat sinks* :: *These guys drank too much* : *These boats sank*] is true, because of the following 4-uple of 6-factorizations, whose factors are aligned column-wise for clarity, and where spaces (underlined) are treated as regular characters:

[2] A solver typically produces several analogical solutions, among which a few are valid.

$$
\begin{aligned}
\mathrm{f}_x &\equiv (\ \mathit{This},\ \ \mathit{\_guy},\ \epsilon,\ \mathit{\_dr},\ \mathit{inks},\ \mathit{\_too\_much}\ ) \\
\mathrm{f}_y &\equiv (\ \mathit{This},\ \mathit{\_boat\_},\ \epsilon,\ \ s,\ \ \mathit{inks},\ \ \ \epsilon\ \ \ \ ) \\
\mathrm{f}_z &\equiv (\ \mathit{These},\ \ \mathit{\_guy},\ \ s,\ \mathit{\_dr},\ \mathit{ank},\ \mathit{\_too\_much}\ ) \\
\mathrm{f}_t &\equiv (\ \mathit{These},\ \mathit{\_boat\_},\ s,\ \ s,\ \ \mathit{ank},\ \ \ \epsilon\ \ \ \ )
\end{aligned}
$$

There is no 4-uple of d-factorizations, with d smaller than 6. Therefore, the degree of this analogy is 6. Note that there are many 4-uple of 6-factorizations that verify the definition, as well as many 4-uple of d-factorizations for d greater than 6.

Although the choice of the definition to work with has some practical impact (the more general the definition, the more complex the machinery to recognize an analogy), we will use the definition given above. Still, the discussion in this chapter generally applies for all sensible definitions we know.

3.2 Searching in the Input Space

Identifying analogies in the input space (step-1) has a cubic complexity with respect to the size of $\mathcal{I}$. Clearly, a brute-force approach would be manageable for toy problems only. This explains why several authors have worked out alternative strategies, as briefly discussed in this section. We refer the reader to [31] for a comparison of those strategies.

A Quadratic Search Procedure The search for input analogies can be transformed into a quadratic number of equation solving [20] thanks to the symmetry property of analogical relations ($[x : y :: z : t] \Leftrightarrow [y : x :: t : z]$). Unfortunately, this solution barely scales to sets of a few thousands of representatives and is not practical: performing analogies on words for realistic NLP applications typically requires to work with vocabularies in the order of 10^5 words. A possible work around is to use sampling techniques.

More precisely, when performing inference for t, we solve analogical equations $[y : x :: i(t) : ?]$ only for some pairs $\langle x, y\rangle$ belonging to the neighborhood of $i(t)$. Solutions belonging to the input space are the z-forms we are looking for. This strategy reduces the search procedure to the resolution of a number of analogical equations which grows quadratically with the size of the neighbor set $\mathcal{N}$:

$$
\begin{aligned}
\mathcal{E}_{\mathcal{I}}(t) = \{\ \langle x, y, z\rangle \mid\ & \langle x, y\rangle \in \mathcal{N}(i(t)) \times \mathcal{N}(i(t)), \\
& [y : x :: i(t) : z]\ \}
\end{aligned}
$$

For instance, in [22] the authors deal with an input space in the order of tens of thousand forms by sampling x and y among the closest neighbors of t, where neighborhood are defined in terms of the standard edit-distance.

Exhaustive Tree-Count Search In [31], the authors developed an alternative approach for scaling up the search procedure. The main idea is to exploit a property of formal analogies [7]:

$$[x : y :: z : t] \Rightarrow |x|_c + |t|_c = |y|_c + |z|_c \quad \forall c \in \Sigma \tag{1}$$

where Σ is the alphabet on which the forms are built, and $|x|_c$ stands for the number of occurrences of character c in x. In the sequel, we denote $\mathcal{C}(\langle x, t\rangle) = \{\langle y, z\rangle \in \mathcal{I}^2 \mid \forall c \in \Sigma, |x|_c + |t|_c = |y|_c + |z|_c\}$ the set of pairs satisfying the count property with respect to $\langle x, t\rangle$.

Their strategy consists in first selecting an x-form in the input space. This enforces a set of necessary constraints on the counts of characters that any two forms y and z must satisfy for $[x : y :: z : t]$ to hold. By considering all forms x in turn,[3] one can collect a set of candidate triplets for t. A verification of those triplets that actually define an analogy with t must then be carried out. Formally, they build:

$$\begin{aligned}\mathcal{E}_{\mathcal{I}}(t) = \{\langle x, y, z\rangle \mid\ & x \in \mathcal{I},\\ & \langle y, z\rangle \in \mathcal{C}(\langle x, i(t)\rangle),\\ & [x : y :: z : i(t)]\}\end{aligned}$$

This strategy will only work if (i) the number of quadruplets to check is much smaller than the number of triplets we can form in the input space (which happens to be the case in practice), and if (ii) we can efficiently identify the pairs $\langle y, z\rangle$ that satisfy a set of constraints on character counts. To this end, the authors proposed to organize the input space thanks to a data structure they call a *tree-count* (hence the name of the search procedure), which is easy to build and supports efficient runtime retrieval.[4]

Sampled Tree-Count Search The tree-count search strategy enables to *exhaustively* solve step 1 for reasonably large input spaces (tens of thousands of forms). However, computing analogies in very large input spaces (hundreds of thousands of forms) remains computationally demanding, as the retrieval algorithm must be carried out $o(\mathcal{I})$ times. In this case, in [31], the authors proposed to sample the x-forms:

$$\begin{aligned}\mathcal{E}_{\mathcal{I}}(t) = \{\ \langle x, y, z\rangle \mid\ & x \in \mathcal{N}(i(t)),\\ & \langle y, z\rangle \in \mathcal{C}(\langle x, i(t)\rangle),\\ & [x : y :: i(t) : z]\ \}\end{aligned}$$

The sampling strategy selects x-forms that share with t some sequences of symbols. To this end, input forms are represented in a κ-dimensional vector space, whose dimensions are frequent symbol n-grams, where $n \in [\texttt{min}; \texttt{max}]$. A form is thus encoded as a binary vector of dimension κ, in which the ith component indicates whether the form contains an occurrence of the ith n-gram. At runtime, the sampling selects the η forms that are the closest to t, according to some distance measure (e.g. the cosine).[5]

[3] Anagram forms do not have to be considered separately.

[4] Possibly involving filtering.

[5] Typical values are $\texttt{min} = \texttt{max} = 3$, $\kappa = 20\,000$, and $\eta = 5\,000$.

Checking for Analogies Some of the aforementioned search strategies require to verify whether each quadruplet in the candidate lists actually defines an analogical relation. Stroppa [32] proposed a dynamic programming algorithm for checking $[x : y :: z : t]$ when the definition in [9] is used. The complexity of this algorithm is in $o(|x| \times |y| \times |z| \times |t|)$. Since a large number of calls to the analogy checking algorithm must be performed during step 1 of analogical learning, the following property may come at help [31]:

$$\begin{array}{l} [x : y :: z : t] \Rightarrow \\ (x[1] \in \{y[1], z[1]\}) \ \vee \ (t[1] \in \{y[1], z[1]\}) \\ (x[\$] \in \{y[\$], z[\$]\}) \ \vee \ (t[\$] \in \{y[\$], z[\$]\}) \end{array} \tag{2}$$

where $s[\$]$ indicates the last symbol of s, and $s[1]$ the first. A simple trick consists in checking the proportionality condition only for quadruplets that pass this test.

Open Issues One can already go a long way with the sampled tree-count approach described above. Still, it is unclear which sampling strategy should be considered for a given application. The vector space model proposed in [31] seems to work well in practice, but more experiments are needed to confirm this.

More fundamentally, none of the search procedures proposed so far takes into account the fact that many analogies might be redundant. For instance, to relate the masculine French noun *directeur* to its feminine form *directrice*, it is enough to consider [*recteur* : *rectrice* :: *directeur* : *directrice*]. Other analogies (i.e. [*fondateur* : *fondatrice* :: *directeur* : *directrice*]) would simply confirm this relation. In [32], Stroppa formalizes this redundancy by the concept of *analogical support set*. Formally, A is an analogical support set of E iff:

$$\{[x : y :: z : ?] : \langle x, y, z \rangle \in A^3\} \supseteq E$$

This raises the question of whether it would be possible to identify a minimal subset of the training set, such that analogical learning would perform equally well in this subset. Determining such a subset would reduce computation time drastically. Also, it would be invaluable for modelling how forms in an input system are related to forms in an output one. We are not aware of studies addressing this issue.

3.3 Solving Equations

Algorithms for solving analogical equations have been proposed for both definitions of interest mentioned above. For the definition of [9], it can be shown [8] that the set of solutions to an analogical equation is a rational language:

Theorem 3.1 *t is a solution to $[x : y :: z : ?]$ **iff** t belongs to $\{y \circ z\} \backslash x$.*

The *shuffle* of two strings w and v, noted $w \circ v$, is the rational language containing all strings obtained by selecting (without replacement) sequences of characters in a

left-to-right manner alternatively in w and v. For instance, *spondyondontilalgiatis* and *ondspondonylaltitisgia* are two strings in the set *spondylalgia* $\circ$ *ondontitis*. The *complementary set* of w with respect to v, noted $w \setminus v$, is the set of strings formed by removing from w, in a left-to-right manner, the symbols in v. For instance, *spondylitis* and *spydoniltis* are belonging to *spondyondontilalgiatis* $\setminus$ *ondontalgia*. This operation can be straightforwardly extended to complement a rational set. The pseudo-code of our implementation of these two operations is provided in Algorithm 1.

```
function shuffle(y,z)
Require: y, and z two forms
Ensure: a random word in y ∘ z
  if y = ϵ then
    return z
  else
    n ← rand(1,|y|)
    return y[1:n] . shuffle(z,y[n+1:])

function complementary(m,x)
Require: m and x, two forms
Ensure: the set m \ x
  return complement(m,x,ϵ)

function complement(m,x,r)
  if (m = ϵ) then
    if (x = ϵ) then
      return {r}
  else
    s1 ← complement(m[2:],x,r.m[1])
    if m[1] = x[1] then
      s2 ← complement(m[2:],x[2:],r)
    return s1 ∪ s2
  return ϕ
```

Algorithm 1: Simulation of the two rational operations required by the solver. $x[a:b]$ denotes the sequence of symbols x starting from index a to index b inclusive. $x[a:]$ denotes the suffix of x starting at index a. `rand`(a, b) returns a random integer between a and b (included).

Since these two operations preserve rationality, it is possible to build a finite-state machine for encoding those solutions In practice however, the automaton is highly non deterministic, and in the worst case, enumerating the solutions can be exponential in the length of the sequences involved in the equation. The solution proposed in [24] consists in sampling this automaton without building it. The more we sample this automaton the more solutions we produce. In our implementation, we call *sampling rate* (ρ) the number of samples in $y \circ z$ that are considered. This is formalized in Algorithm 2.

It is important to note that typically, a solver produces several solutions to an equation, most of them spurious, which means that even though they obey the definition of formal analogy, they are not linguistically valid. To illustrate this, Fig. 1 reports solutions produced to the equation [*even* : *usual* :: *unevenly* : *?*] by our implementation of Algorithm 2. As can be observed, most solutions are not valid forms in English: indeed, this definition recognizes no less than 72 different legitimate solutions, which we were able to produce with enough sampling ($\rho =$ 2000) in less than a few tenth of a millisecond.[6]

[6] The time measurements reported in this study have been made on an ordinary desktop computer, and are provided for illustration purposes only.

function `solver`($\langle x, y, z\rangle$, ρ)
Require: $\langle x, y, z\rangle$, a triplet, ρ the sampling rate
Ensure: a set of solutions to $[x : y :: z : ?]$
  $sol \leftarrow \phi$
  for $i \leftarrow 1$ to ρ **do**
    $\langle a, b\rangle \leftarrow$ `odd(rand(0, 1))` ? $\langle z, y\rangle : \langle y, z\rangle$
    $m \leftarrow$ `shuffle`(a, b)
    $c \leftarrow$ `complementary`(m,x)
    $sol \leftarrow sol \cup c$
  return sol

Algorithm 2: A Stroppa & Yvon flavored solver. `rand`(a, b) returns a random integer between a and b (included). The ternary operator `?:` is to be understood as in the C language.

ρ	nb	solutions		
20	12	*usuaunlly* (3)	***unusually*** (2)	*usunually* (2)
100	34	***unusually*** (6)	*usuaunlly* (6)	*uunsually* (4)
1000	67	***unusually*** (57)	*uunsually* (23)	*usuunally* (19)
2000	72	***unusually*** (130)	*uunsually* (77)	*usunually* (43)

Fig. 1 3-most frequent solutions to [*even* : *usual* :: *unevenly* : *?*] along with their frequency, as produced by our solver, as a function of the sampling rate ρ. *nb* stands for the total number of solutions produced

The problem of multiple solutions to an equation is exacerbated when we deal with longer forms. In such cases, the number of spurious solutions can become quite large. As a simple illustration of this, consider the equation $e =$ [*this guy drinks too much* : *this boat sinks* :: *those guys drink too much* : *?*] where forms are considered as strings of characters (the space character does not have any special meaning here). Figure 2 reports the number of solutions produced as a function of the sampling rate. For small values of ρ, the solution might be missed by the solver (i.e. $\rho \leq 20$). For larger sampling rates, the expected solution typically appears (with frequent exceptions) among the most frequently generated ones. Note that the number of solutions generated also increases drastically. Clearly, enumerating all the solutions is not a good idea: too many solutions, too time consuming.

The fact that a solver can (and typically does) produce spurious solutions means that we must devise a way to distinguish "good" solutions from spurious ones. We defer this issue to the next section. Yet, we want to stress that currently, our sampling of the automaton that recognizes the solutions to an equation is completely random. It would be much more efficient to learn to sample the automaton, such that more likely solutions are enumerated first. Several algorithms might be applied for this task, among which the Expectation-Maximization algorithm for transducers described in [33].

$\rho = 20$	$nb = 8$
$t = 0.0003$	$r = \phi$
thos_boate_sinks (2)	
tho_boatse_sinks (2)	
thoboatse__sinks (2)	

$\rho = 100$	$nb = 28$
$t = 0.001$	$r = 13$
thoboatse__sinks (2)	
tho_boatse_sinks (2)	
those_sboat_sink (2)	

$\rho = 1000$	$nb = 28$
$t = 0.009$	$r = 2$
those_boat_ssink (5)	
those_boats_sink (5)	
thoes_tboa_sinks (5)	

$\rho = 10^6$	$nb = 19\,796$
$t = 3.82$	$r = 10$
thoes_boat_sinks (2550)	
thoses_boat_sink (1037)	
those_boat_ssink (999)	

Fig. 2 3-most frequent solutions produced by our solver at different sampling rates for the equation *e*. `r` indicates the position of the expected solution in the list if present (ϕ otherwise). *nb* indicates the number of solutions produced, and `t` the time counted in seconds taken by the solver. For readability, spaces are represented with the symbol _.

3.4 Aggregating Solutions

Step-3 of analogical learning consists in aggregating all the solutions produced. We saw in the previous section that the number of solutions to an analogical equation can be quite large. Also, there might be quite a large number of analogical equations to solve during step-2, which simply increases the number of solutions gathered in $\mathcal{E}_o(t)$. In many works we know, this problem is not discussed; yet, our experiments indicate that this is a important issue. In [34], Lepage and Lardilleux filter out solutions containing sequences of symbols not seen in the output space of the training set. This typically leaves many solutions alive, including spurious ones. In [20], Lepage and Denoual propose to keep the most frequently generated solutions. The rationale being that forms that are generated by various analogical equations are more likely to be good ones. Also, Ando and Lepage [35] show that the closeness of objects in analogical relations is another interesting feature for ranking candidate solutions.

In [24], the authors investigate the use of a classifier trained in a supervised fashion to recognize correct solutions from bad ones. This approach improved the selection mechanism over several baselines (such as selecting the most frequently generated solution), but proved to be difficult to implement, in part because many examples have to be classified, which is time consuming, but also because most of the solutions in $\mathcal{E}_o(t)$ are spurious ones, leaving us with a very unbalanced task, which is challenging. Last but not least, the best classifiers trained were using features computed on the whole set $\mathcal{E}_o(t)$, such as the frequency with which a solution is proposed. This means that it cannot be used to filter the unlikely solutions generated for a test form t in the early stages.

Improving the classifier paradigm deserves further investigations. Notably, in [24], only a small number of features have been considered. Better feature engineering, as well as more systematic tests on different tasks must be carried out to better understand the limitations of the approach.

As discussed in [35], it is intuitively more suited to see the problem of separating correct from spurious solutions as a ranking problem. Ranking is an active research topic in machine learning. We refer the reader to the LETOR (LEarning TO Rank) website for an extensive list of resources on this subject.[7] Ranking the solutions proposed by the two-first steps of analogical learning must be investigated as a replacement of the classification solution proposed in [24].

3.5 Dealing with Silence

In most experiments we conducted, we faced the problem that the learning mechanism might fail to produce a solution for a given test form. This can happen because no input analogy is found, or because the input analogies identified yield output equations that have no solution. Depending on the nature of the input space and the training material available, this problem can be rather important.

On a task of translating medical terms [24], the authors submitted the silent cases to another approach (in their case a statistical translation engine). Combining analogical learning with statistical machine translation has also been investigated in [36]. In [20], the authors proposed to split the form to treat in two parts and apply analogical learning to solve those two subforms. This raises a number of issues which do not seem to have received a lot of attention. Knowing where to split the input form in order to maximize the chance of being able to solve the two new sub-problems is one of those.

4 Case Study

4.1 Settings

In order to illustrate some of the elements discussed in previous sections, we applied analogical learning to the task of transliterating English proper names into Chinese. The task we study is part of the NEWS evaluation campaign conducted in 2009 [37]. Transliteration is generally defined as the phonetic transcription of names across languages, and is often thought as a critical technology in many domains, such as machine translation and cross-language information retrieval or extraction [37].

Examples of transliterations from English proper names into Chinese are reported in Table 1. The segmentations (represented by + signs) are ours, and were produced by inspection of the English-into-Chinese transcription table available

[7] http://research.microsoft.com/en-us/um/beijing/projects/letor/.

Table 1 Examples of transliterations of English proper names into Chinese taken from the NEWS 2009 English-Chinese corpus

A+co+ca	阿+克+卡
A+f+ri+ca+nu+s	阿+弗+里+卡+纳+斯
A+lessan+d+ro+ni	亚+历山+德+罗+尼
Bo+lan+d	伯+兰
Bo+ree+l	伯+雷+尔
B+ra+d+fo+rd	布+拉+德+福+德
C+ro+sse+tt	克+罗+西+特
Na+v+ra+ti+lo+va	纳+夫+拉+蒂+洛+娃
Ne+me+ro+v	内+梅+罗+夫

The segmentations (shown with + signs) are ours, and are intended to illustrate the matching between English and Chinese sequences of characters

Table 2 Main characteristics of the English-into-Chinese data provided in NEWS 2009

	train	*dev*	*train* + *dev*	*test*
Examples	31961	2896	34857	2896
EN symbols	52	51	52	51
CH symbols	370	275	371	283
Avr. EN-length	6.8	6.8	6.8	6.9
Min EN-length	2	3	2	3
Max EN-length	15	15	15	13
Avr. CH-length	9.5	9.5	9.5	9.6
Min CH-length	3	3	3	3
Max CH-length	21	21	21	21

on Wikipedia.[8] As can be observed, the transcription is written with monosyllabic characters which may not correspond exactly to syllables in English. A Chinese character may correspond to different English ones, and vice versa. For instance, the character *A* is transcribed into 阿 in *Acoca*, but not in *Alessendroni* (where the sound "ya" is assumed). Note also that special rules are used to encode initial characters, therefore, it is not advisory to convert English names into lower case, as was done in [38]. Also, a transcription sometimes reflects the meaning as well as the sound of the transcribed word. For instance, the common ending *-va* in Slavic female family names is usually transcribed as 娃 (girl), as in *Navratilova*.

The organizers of the NEWS campaign kindly provided us with the data that was distributed to the participants of the English-into-Chinese transliteration task. Its main characteristics are reported in Table 2. The distribution of Chinese characters is typically Zipfian, and 116 out of the 370 different characters seen in the training set appear less than 10 times (39 characters appear only once).

In order to transliterate the English proper names of the test set, we gathered a training set $\mathcal{L}_1 = train + dev$ by concatenating the training set and the development

[8] http://en.wikipedia.org/wiki/Template:Transcription_into_Chinese. We could not segment *lessan* with this table.

set that were released, that is, 34,857 pairs of English and Chinese proper names. Including the development set in the training material is fine, since there is no training involved when generating the set of solutions. In parallel to this, we also generated solutions for the development set (*dev*), using the training material only ($\mathcal{L}_2 = train$); the solutions produced were used for training a classifier to recognize good from spurious solutions. This classifier was then applied to the solutions produced for the test set (*test*) thanks to $\mathcal{L}_1$.

We mainly ran two configurations of our *generator*. The first one, named FULL- TC, corresponds to the exhaustive tree-count setting described in Sect. 3.2. The second one, named SAMP- TC, corresponds to the sampled version described in Sect. 3.2, where the $\eta = 1{,}000$ closest input forms to each English test form were considered, based on a vector space representing the $\kappa = 5{,}000$ most frequent 3-grams of characters observed in $\mathcal{I}$, and the cosine distance. In both cases, the solver was run with a sampling rate of $\rho = 400$. We decided to keep up to the 100-most frequently generated solutions for a given test form. A solution is typically generated several times per equation (the frequency of which depends of ρ), and by several equations.

Regarding the classifier, we followed [24] and trained a voted-perceptron [39]. We computed a total of 33 (partly redundant) features including the frequency of a solution, its rank in the list, average degrees of the input and output analogies leading to a solution, likelihoods of several n-gram language models trained at the (Chinese) character level, etc. We trained the classifier over 5,000 epochs. A greedy search over the feature set revealed that half of these features are enough for optimal performance.

4.2 Monitoring Analogical Inference

In the following, we describe in details the FULL- TC configuration, while Table 3 reports the figures of interest for both configurations we tested. For the exhaustive configuration, most of the time is spent during the search for input analogies. Roughly 8 s is spent on average per input form in order to identify an average of 4,097 analogies (and a maximum of 47,176 ones). For some forms, the time for identifying input analogies can be as long as 37 s. This suggests a timeout strategy when too many candidate analogies are observed. Also, it is likely not necessary to consider all input analogies, which raises the issue of ranking analogies to be treated first. For 37 of the test forms, we could not identify any input analogy, leading to no response in those cases. Solving all output equations led to an average of 278 solutions per test form (minimum 0, maximum 1,918). Note that many equations solved led to no solution, which explains why on average, the number of solutions is lower than the number of equations solved. The average time for solving the equations per form was 0.7 s, with a worst case of 7.3 s (because many equations needed to be solved). It is interesting to note the discrepancy between the number of input analogies identified and the number of output equations effectively solved, which is much lower. This indicates either that the input analogies were in large part fortuitous, or that the inference bias

Table 3 Details of the two configurations tested

	FULL- TC			SAMP- TC		
	Avr.	Min.	Max.	Avr.	Min.	Max.
Candidates analogies	42122	1	544849	1374	0	14657
Candidates to check	30629	0	381762	1001	0	0595
Avr. input analogies	4097	0	47176	179	0	1223
Output equations	512	0	47176	28	0	239
Solutions	278	0	1918	40	0	402
w/o input analogy	37 (1.3 %)			114 (3.9 %)		
w/o solution	69 (2.4 %)			223 (7.7 %)		
Locate time	4.2 s	0.7 s	11.6 s	0.02 s	0	0.7 s
Check time	3.6 s	0	25.5 s	0.03 s	0	0.3 s
Solving time	0.7 s	0	7.3 s	0.01 s	0	0.1 s
Total time	8.5 s	0.9 s	38.3 s	0.05 s	0	0.7 s

(one analogy in the input space corresponds to an analogy in the output space) does not apply well for this task. In the end, our inference procedure remained silent for 106 input forms (3.7 %).

As expected, the SAMP- TC configuration runs much faster, identifying far less input analogies (179 on average) and leaving 337 input forms (11.6 %) without answer. This shows that while effective at reducing computation time, the tree-count search strategy described in Sect. 3.2 is perfectible.

4.3 Evaluation

In this section, we analyze the different variants of the analogical devices we developed. We start by a detailed analysis of the core variants that were tested, followed by a broader evaluation conducted thanks to the evaluation scripts available on the NEWS 2009 website. In both cases, let us consider a transliteration experiment as a set of N (here, $N = 2896$) triplets, $(e_i, r_i, a^i_{1,\ldots,n_i} \equiv \{c^i_1, \ldots, c^i_{n_i}\})_{i\in[1,N]}$, where e_i is the ith test form, r_i its reference transliteration,[9] and $a^i_{1,\ldots,n_i}$ the ranked list of the n_i (here, $0 \leq n_i \leq 100$) solutions proposed by analogical learning, where solutions are ranked in decreasing order of likeliness.

Detailed Evaluation Let n_k be the number of test forms for which the reference solution is seen in the k-first solutions proposed: $n_k \equiv \sum_{i=1}^{N} \delta(\{r_i\} \cap a^i_{1,\ldots,k} \neq \phi)$, where $\delta(x)$ is the Kronecker function the value of which is 1 if x is true, and 0 otherwise. We define $ratio_k$ as the ratio of test forms with a sanctioned solution ranked in the k-first solutions, to the number of test forms with a sanctioned solution.

[9] For the English-into-Chinese transliteration task we consider, there is only one reference for each test form.

Table 4 Quality of the 100-top frequent solutions proposed by the FULL- TC and the SAMP- TC configurations

	FULL- TC				SAMP- TC			
		$ratio_k$	$prec_k$	rec_k		$ratio_k$	$prec_k$	rec_k
k	nb	(%2377)	(%2790)	(%2896)	nb	(%1693)	(%2559)	(%2896)
1	1258	52.9	45.1	43.4	792	46.8	28.4	27.4
2	1548	65.1	55.5	53.5	1046	61.8	37.5	36.1
3	1706	71.8	61.2	58.9	1207	71.3	43.3	41.7
4	1819	76.5	65.2	62.8	1328	78.4	47.6	45.9
5	1904	80.1	68.2	65.8	1420	83.9	50.9	49.0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
26	2304	96.9	82.6	79.6	1693	100.0	60.7	58.5
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
86	2377	100.0	85.2	82.1	1693	100.0	60.7	58.5

nb indicates the number of correct transliterations at a given rank k. Read the text for more

We define $prec_k$ as the percentage of test forms with at least k candidate translations out of which at least one is a sanctioned solution and rec_k the percentage of test forms with a sanctioned transliteration proposed in the k-first positions. Formally:

$$\begin{aligned} ratio_k &= n_k / \textstyle\sum_{i=1}^{N} \delta(\{r_i\} \cap a^i_{1,\ldots,n_i} \neq \phi) \\ prec_k &= n_k / \textstyle\sum_{i=1}^{N} \delta(n_i > 0) \\ rec_k &= n_k / N \end{aligned}$$

Intuitively, $prec_k$ is a measure of precision, and rec_k a measure of recall, while *ratio* provides the distribution of the rank of the correct solutions identified. Those figures (expressed as percentage) are reported in Table 4 for the two configurations of the generator we tested: FULL- TC and SAMP- TC. The first line of the first column indicates for instance that 45.1 % of the test forms with at least one solution have the sanctioned transliteration produced in the first position. This represents 52.9 % of the test forms with a sanctioned solution, and 43.4 % of the total test forms.

This table calls for several comments. We noted earlier that the inference procedure leaves some forms without solution (3.7 % for FULL- TC and 11.6 % for SAMP- TC). Furthermore, we observe that the recall is reaching a limit of 82.1 % for FULL- TC and 58.5 % for SAMP- TC. Therefore, there is a fair number of cases which are not handled appropriately by the inference mechanism. For the SAMP- TC configuration, this simply shows the shortcomings of a too aggressive sampling strategy. For the FULL- TC variant, and while part of this recall failure can be explained by the fact that we only consider the 100-most frequent solutions generated, it simply illustrates the silence issue discussed in Sect. 3.

Table 5 Impact of the classifier applied in the aggregation step of the FULL- TC configuration

	FULL- TC + classifier			
		$ratio_k$	$prec_k$	rec_k
k	nb	(%1667)	(%2790)	(%2896)
1	**1577** (↑ 319)	**94.6** (↑ 41.8)	**56.5** (↑ 11.4)	**54.5** (↑ 11.1)
2	**1652** (↑ 104)	**99.1** (↑ 34.0)	**59.2** (↑ 3.7)	**57.0** (↑ 3.5)
3	1661 (↓ 45)	99.6 (↑ 25.1)	59.5 (↓ 1.7)	57.4 (↓ 1.5)
4	1662 (↓ 157)	99.7 (↑ 23.2)	59.6 (↓ 5.6)	57.4 (↓ 5.4)
5	1662 (↓ 242)	99.7 (↑ 19.6)	59.6 (↓ 8.6)	57.4 (↓ 8.4)
⋮	⋮	⋮	⋮	⋮
16	1667 (↓ 545)	100.0 (↑ 6.9)	59.7 (↓ 19.6)	57.6 (↓ 18.8)

Considering only the most frequently generated solution,[10] recall drops to 43.4 and 27.4 % respectively. This shows that being able to distinguish good from spurious solutions has the potential to improve the overall approach by more than 30 absolute points in both configurations.

The effect of applying the classifier described above is analyzed in Table 5 for the FULL- TC configuration (the most effective one). It is noticeable that the number of test forms with the sanctioned solution ranked first increases significantly, which is good: another 319 forms are ranked first, which translates into an increase of precision and recall at rank 1 of above 11 absolute points. Precision and recall at rank 2 also gain over 3 absolute points. This illustrates that classifying solutions is feasible and leads to better performance than simply picking the most frequently generated form. It must be noted however, that this gain comes at the cost of precision and recall at higher ranks: some correct solutions are actually being removed by our classifier.

NEWS 2009 Table 6 reports the results of several of the transliteration devices we devised, as measured by the official metrics of the NEWS 2009 evaluation campaign [37]. Since the MAP_{ref} metric reported always equals ACC in our experiments, we only report the three first metrics output by the evaluation script[11] we used: ACC which corresponds to rec_1,[12] F-score which gives a partial credit proportional to the longest subsequence between the reference transliteration and the first candidate one, and the Mean Reciprocal Rank (MRR), where 100/MRR roughly indicates the average rank of the correct solution over the session. See [37] for the details of their computation. Those figures are expressed as percentage.

Table 6 also calls for several comments. First, we tested different variants of the SAMP- TC strategy (lines 1–6). The dimension κ of the space into which the (input)

[10] In the SAMP- TC configuration, 20 test forms receive several solutions with the same maximum frequency, among which the sanctioned transliteration. We do credit the system with a rank 1 solution in those cases, even if the correct transliteration is not listed first.

[11] We downloaded the script `news_evaluation.py` at http://translit.i2r.a-star.edu.sg/news2009/evaluation/.

[12] To the exception of the way we broke ties in the unfrequent cases where several solutions are produced with the top frequency.

Table 6 Evaluation of different configurations with the metrics used at the NEWS 2009 evaluation campaign

	Configurations		ACC	F-score	MRR	No sol.
1	SAMP- TC	$\eta = 1\text{k}, \kappa = 5\text{k}$	26.7	56.7	34.7	337 (11.6%)
2	SAMP- TC	$\eta = 1\text{k}, \kappa = 10\text{k}$	30.6	60.2	39.1	268 (9.3%)
3	SAMP- TC	$\eta = 2\text{k}, \kappa = 10\text{k}$	33.7	63.1	42.5	202 (7.0%)
4	SAMP- TC	$\eta = 5\text{k}, \kappa = 10\text{k}$	36.7	66.3	46.1	148 (5.1%)
5	SAMP- TC	$\eta = 10\text{k}, \kappa = 10\text{k}$	39.4	68.2	49.0	120 (4.1%)
6	SAMP- TC	$\eta = 15\text{k}, \kappa = 10\text{k}$	41.0	69.2	50.5	110 (3.8%)
7	FULL- TC		43.4	70.5	52.4	106 (3.7%)
8	FULL- TC	$d \leq 4$	46.6	72.6	55.7	173 (6.0%)
9	FULL- TC	$d \leq 3$	49.7	74.0	57.6	265 (9.2%)
10	FULL- TC	$d \leq 2$	55.1	73.4	61.4	411 (14.2%)
11	Cascading(10,9,8,7)		56.4	79.2	62.9	106 (3.7%)
12	FULL- TC	+ VP_{all}	53.3	77.0	54.5	106 (3.7%)
13	FULL- TC	+ VP_{subset}	54.5	77.6	55.9	106 (3.7%)
14	Cascading(10*, 9*, 8*, 10, 9, 8, 7*, 7)		57.0	79.5	61.7	106 (3.7%)
15	Last NEWS 2009		19.9	60.6	22.9	–
16	First NEWS 2009		73.1	89.5	81.2	–

For comparison, *first* and *last* indicate the first and last performing systems respectively, as reported in [37]. In the cascading configurations (lines 11 and 14), components are listed in order in which they are consulted; those marked with a * sign are using the VP_{all} classifier. See text for further explanations

forms are represented is of importance, and setting κ to 10,000 (line 2) improves the accuracy by almost 4 absolute points compared to setting it to 5,000 (line 1), as we did in the previous section. Increasing κ further does not help, but ultimately, this parameter should be adjusted for each task. Naturally, considering a larger number (η) of neighbors leads to better performance, as shown by lines 2–6. Still, considering all the input forms (line 7) yields the best performance.

Second, we tested the influence of the maximum degree accepted for an analogy (input or output), which we call d-limit in the sequel. Considering only degree 2 analogies (line 10) yields a much higher score than considering all analogies, and this, even though the number of unsolved test forms increases. The optimal d-limit should be adjusted for each task. For instance, in their experiments on unsupervised morphology acquisition, Lavallée and Langlais [14] found that a d-limit of 5 was optimal for the German language, while a d-limit of 2 was enough for capturing the English morphology. Since identifying low degree analogies can be implemented more efficiently, and seems to produce correct solutions here, this suggests a cascading strategy where degree-2 analogies are searched first. Then, degree-3 analogies are searched for the only test forms for which no solution has been provided yet, and so on, until the d-limit (if any) is met. This is simulated in line 11 of Table 6. By doing so, we increase the recall of analogical inference, which benefits the overall performance of the device.

Third, we tested the influence of the classifier during the aggregation step. We trained and tested different flavors of the voted-perceptron algorithm, varying the

feature representation, as well as the number of epochs. We report two of these variants in lines 12 and 13 of Table 6. The first one has been trained on the totality of the features we computed (VP_{all}) and leads to a performance (ACC) of 53.3 %, while the second one obtains 54.5 % by using a subset of the features only (VP_{subset}).[13] Note that for these two variants, when all the solutions of a test form are being pruned by the classifier, we keep the one with the largest classification score. The classifier trained on less features outperforms the other, showing that feature selection should be considered for optimal performance. In both cases, it is important to note that the gain in accuracy is huge compared to the FULL- TC configuration in line 7 (above 10 absolute points in accuracy), which shows the importance of the aggregation step.

Fourth, we tested a last configuration by cascading 8 variants: the ones described in lines 7 to 10, that is, the FULL- TC configuration with varying d-limits, and their variants using a classifier. While we could have train a classifier specific to each variant, we used the same classifier here (the one with all the features, VP_{all}) for all the aggregation steps. The order of the cascading reflected the intuition that classified variants should be consulted first (since their precision is expected to be higher), in the order of their d-limit (2, then 3, etc.). The results are reported in line 14 and show a slight improvement over the cascading configuration reported in line 11.

Last, we observe that our best system is not among the leading ones at NEWS 2009 (see line 16); 18 systems participated to the 2009 English-into-Chinese transliteration exercise. In fact, we would have ended up at the 12th rank according to accuracy (ACC), the official metric at NEWS.

Since our major goal was to monitor analogical learning, we did not put efforts into improving those figures, although there are straightforward things that could be done, such as always providing 10 candidate solutions, even if the classifier filtered in much less (except for accuracy, the other metrics are assuming a list of 10 candidates). Also, we did not attempt anything for dealing with silent test forms. In [36], the authors show that combining in a simple way analogical learning with statistical machine translation can improve upon the performance of individual systems. Last, it is shown in [36] that representing examples as sequences of syllables instead of characters (as we did here) leads to a significant improvement of analogical learning on a English-into-Indi transliteration task.

4.4 Transliteration Session

As an example, we illustrate in Figs. 3 and 4 how the name *Zemansky* was transliterated into 泽曼斯基.[14] 1 247 candidate analogies were identified thanks to the tree-count strategy, 405 of them were filtered thanks to the property defined by Eq. 2. Verifying the 842 remaining candidates took 0.15 s, and 53 (input) analogies were finally identified. After projection, this led to 7 (output) equations with at least

[13] 17 features were considered, based on a greedy search over the feature space, minimizing the training error rate over 100 epochs.

[14] We took this example just because the number of analogies involved is small enough.

Sch	ell	→	谢	尔		U	linski	→	乌	林	斯 基
Z	ell	→	泽	尔		U	mansky	→	乌	曼	斯 基
Sch	emansky	→	谢	曼	斯 基	Ze	linski	→	泽	林	斯 基
Z	emansky	→	泽	曼	斯 基	Ze	mansky	→	泽	曼	斯 基

Fig. 3 Details of the two productive pairs of analogies involved in solving *Zemansky*. Factors that commute are aligned vertically for lisibility

one solution. Those equations, together with the input analogies that fired them are reported in the figure. Only two equations yielded the sanctioned transliteration; both involving analogies of degree 2. All together, this session took slightly less than 4 s, mostly spent for searching the (input) candidates.

The two pairs of analogies yielding the sanctioned transliteration are illustrated in Fig. 3. We observe that the first pair of analogies passively captures the fact that the two substrings *Sch* and *Z* correspond respectively to 谢 and 泽. Similarly, the second pair of analogies somehow captures that the substrings *U* and *Ze* correspond respectively to 乌 and 泽. However, there is no attempt to learn such a mapping.

5 Discussion

In this study, we presented in Sect. 1 a number of works on formal analogy dedicated to various NLP tasks. We described in Sect. 2 the analogical inference procedure and discussed in Sect. 3 a number of issues that we feel remain to be investigated for the approach to meet higher acceptance among the NLP community. In particular, we presented a number of approaches for tackling large input spaces, and discussed their limitations. We pointed out the noise of the inference procedure that must be taken care of. We also noted that the inference procedure suffers a recall problem for which very few solutions have been proposed. In Sect. 4, we presented a case study, transliteration of proper names, for which we reported encouraging results. More importantly, we used this case study for illustrating some of the issues behind the scene of analogical learning.

We believe that analogical inference over sequences has not delivered all its potential yet. We already discussed a number of issues that remain to be investigated. In particular, our experiments in Sect. 4 show that reducing the time complexity of the search for analogies without impacting performance is still challenging. Identifying low degree analogies first might be of help here, since they are easier to spot. We also observed that there is a large room for improvements in the aggregation step. The experiments we conducted with a classifier trained to recognize correct analogies in this study have just scratched the surface. We also discussed the problem of silence that the approach meet in typical NLP applications, including transliteration we visited in this work. Here again, low degree analogies could be put at use in order to guide the split of forms in parts that analogical inference can handle.

We are currently investigating three follow up issues. First, we are engineering a much larger feature set than the one we considered in this work. In particular, we are

31 solutions:

泽 曼 斯 基 (f=77)	泽 门 斯 基 (f=59)	齐 斯 曼 基 (f=29)	曼 泽 斯 基 (f=29)
齐 斯 基 曼 (f=22)	兹 梅 斯 卡 尼 (f=20)	泽 门 基 斯 (f=11)	泽 斯 门 基 (f=9)
齐 曼 斯 基 (f=9)	泽 曼 基 斯 (f=8)	曼 斯 泽 基 (f=8)	泽 斯 曼 基 (f=8)
兹 梅 斯 尼 卡 (f=8)	兹 梅 尼 斯 卡 (f=6)	泽 基 门 斯 (f=5)	兹 尼 梅 斯 卡 (f=4)
齐 基 斯 曼 (f=4)	泽 斯 基 门 (f=4)	斯 泽 门 基 (f=3)	斯 齐 基 曼 (f=3)
斯 基 齐 曼 (f=2)	曼 泽 基 斯 (f=2)	泽 基 曼 斯 (f=2)	泽 斯 基 曼 (f=2)
斯 齐 曼 基 (f=2)	尼 兹 梅 斯 卡 (f=2)	曼 斯 基 泽 (f=2)	斯 基 泽 门 (f=1)
基 齐 斯 曼 (f=1)	基 泽 门 斯 (f=1)	斯 泽 基 门 (f=1)	

[*Stephens* : *Stephansky* :: *Zemens* : *Zemansky*] (d=2)
↪ [斯 蒂 芬 斯 : 斯 蒂 芬 斯 基 :: 泽 门 斯 : ?] (9 solutions)

斯 泽 基 门 (d=6)	泽 基 门 斯 (d=3)	泽 斯 门 基 (d=4)
基 泽 门 斯 (d=3)	斯 泽 门 基 (d=4)	泽 斯 基 门 (d=4)
泽 门 基 斯 (d=3)	斯 基 泽 门 (d=4)	泽 门 斯 基 (d=2)

[*Rovine* : *Rovensky* :: *Zieman* : *Zemansky*] (d=2)
↪ [罗 文 : 罗 文 斯 基 :: 齐 曼 : ?] (6 solutions)

斯 基 齐 曼 (d=3)	斯 齐 曼 基 (d=4)	斯 齐 基 曼 (d=5)
齐 曼 斯 基 (d=2)	齐 斯 曼 基 (d=4)	齐 斯 基 曼 (d=3)

[*Schell* : *Zell* :: *Schemansky* : *Zemansky*] (d=2)
↪ [谢 尔 : 泽 尔 :: 谢 曼 斯 基 : ?] (4 solutions)

曼 斯 泽 基 (d=4)	泽 曼 斯 基 (d=2)
曼 斯 基 泽 (d=4)	曼 泽 斯 基 (d=4)

[*Beale* : *Zmeskal* :: *Beaney* : *Zemansky*] (d=8)
↪ [比 尔 : 兹 梅 斯 卡 尔 :: 比 尼 : ?] (5 solutions)

兹 梅 斯 尼 卡 (d=4)	兹 尼 梅 斯 (d=4)	兹 梅 尼 斯 卡 (d=4)
尼 兹 梅 斯 卡 (d=4)	兹 梅 斯 卡 尼 (d=2)	

[*Clise* : *Zisman* :: *Cleskey* : *Zemansky*] (d=6)
↪ [克 莱 斯 : 齐 斯 曼 :: 克 莱 斯 基 : ?] (8 solutions)

斯 齐 基 曼 (d=5)	斯 齐 曼 基 (d=6)	斯 基 齐 曼 (d=5)
基 齐 斯 曼 (d=3)	齐 基 斯 曼 (d=3)	齐 斯 基 曼 (d=3)
齐 斯 曼 基 (d=4)	齐 曼 斯 基 (d=4)	

[*Heale* : *Zmeskal* :: *Heaney* : *Zemansky*] (d=8)
↪ [希 尔 : 兹 梅 斯 卡 尔 :: 希 尼 : ?] (5 solutions)

兹 梅 斯 尼 卡 (d=4)	兹 尼 梅 斯 卡 (d=4)	兹 梅 尼 斯 卡 (d=4)
尼 兹 梅 斯 卡 (d=4)	兹 梅 斯 卡 尼 (d=4)	

[*Ulinski* : *Umansky* :: *Zelinski* : *Zemansky*] (d=2)
↪ [乌 林 斯 基 : 乌 曼 斯 基 :: 泽 林 斯 基 : ?] (9 solutions)

泽 斯 曼 基 (d=5)	曼 泽 斯 基 (d=5)	曼 斯 基 泽 (d=4)
泽 斯 基 曼 (d=5)	泽 基 曼 斯 (d=5)	曼 斯 泽 基 (d=4)
曼 泽 基 斯 (d=7)	泽 曼 基 斯 (d=5)	泽 曼 斯 基 (d=2)

Fig. 4 Log of the FULL- TC transliteration session for the English proper name *Zemansky*. 31 solutions have been identified; the one underlined (actually the most frequently generated) is the sanctioned one. (f=•) indicates the frequency of a solution, while (d=•) indicates the degree of an analogy

considering features extracted from often co-occurring pairs of (English/Chinese) sequences of symbols. This raises drastically the dimensionality of the representations given to the classifier, for hopefully better discriminative power. Second, we are investigating different machine learning algorithms. Preliminary results indicate that support vector machines [40] slightly outperform the voted perceptron algorithm we used in this study. Also, we are testing a number of rescoring approaches that seem to lead to higher overall performance. Further investigations are needed to be conclusive. Last but not least, we mentioned that our equation solver produces many (spurious) solutions. Filtering them out a posteriori, as we tried in this study, is fruitful even though the classifier we trained is far not perfect. But, one may legitimately argue that a better solution consists in fixing the solver in the first place. Currently, our solver ranks the solutions it produces in decreasing order of frequency. Some preliminary work we conducted on solving analogical equations among English proper names show that ranking the solutions with a language model is a much better solution (the correct solution is ranked higher in the list of solutions). This observation, which likely depends on the domain investigated as well as the language considered, at the very least suggests that finding ways of guiding the solver to produce less, but promising solutions is feasible.

Learning to solve equations is therefore a promising avenue that we plan to address. Currently, our solver randomly reads out paths of the non deterministic finite-state automaton which recognizes all possible solutions to an equation, according to the definition of formal analogy given in Sect. 3.1. While learning to weight its arcs can be tempting (see for instance the algorithm described in [41]), we must not forget that each equation leads to a particular automaton, and that building the union of all possible such automata is not realistic, which seriously questions the applicability of a learning algorithm that assumes a given topology as input. One way to tackle the problem consists in discriminatively learn a function the features of which would depend on some characteristics of each automaton (for instance features based on n-gram sequences), and using this function to guide the sampling of paths in the automaton. Another way of addressing the issue of producing less but better solutions is to introduce some knowledge directly into the solver. In a way, this is what Lepage [42] did with his solver, which produces only a subset of the solution that our solver would generate. Of course, introducing some knowledge into the solver raises the issue of its generality (to domains and to languages). This clearly deserves investigations.

While we concentrated in this study on analogical learning over sequences of symbols, it should be stressed that this learning paradigm is not limited to this scenario only. Stroppa [32] shows it can be generalized to structures of interest in NLP such as trees and lattices. In particular, he proposed a definition of formal analogy on trees, based on the notion of factorization of trees, very much in line with the definition of formal analogies between sequences of symbols defined in [9]. Based on this definition, the authors of [10] described an exact algorithm for solving an analogical equation on trees which complexity is at least exponential in the number of nodes of the largest tree in the equation. They also proposed two approximate solvers by constraining the type of analogies captured (notably, passive/active alternations are

not anymore possible). Ben Hassena [12] proposed a solution for reasoning with trees based on tree alignment. The constraints imposed over the possible alignments are much more restrictive than the ones of [10], but the author reports a solver (a dynamic programing algorithm) which has a polynomial complexity. Unfortunately, none of the aforementioned approaches scale to even medium-sized corpora of trees. For instance in [12] the author applied analogical learning on a training set of less than 300 tree structures, a very small corpus by today's standards. See also the work of Ando and Lepage [35] for a very similar setting.

Acknowledgments This work has been partially founded by the Natural Sciences and Engineering Research Council of Canada (NSERC). We are grateful to the anonymous reviewers of the short paper submitted to the 2012 SAMAI workshop, as well as those that reviewed this article. We found one review in particular especially inspiring.

References

1. Turney, P., Littman, M.: Corpus-based learning of analogies and semantic relations. Mach. Learn. **60**, 251–278 (2005)
2. Duc, N.T., Bollegala, D., Ishizuka, M.: Cross-language latent relational search: mapping knowledge across languages. In: AAAI'11, pp. 1237–1242 (2011)
3. Marshall, J.B.: A self-watching model of analogy-making and perception. J. Exp. Theor. Artif. Intell. **18**(3), 267–307 (2002)
4. Hofstadter, D.R.: The Copycat Project: An Experiment in Nondeterminism and Creative Analogies, vol. 755. Massachusetts Institute of Technology, Cambridge (1984)
5. Mitchell, M.: Analogy-Making as Perception. MIT Press/Bradford Books, Cambridge (1993)
6. Yvon, F.: Paradigmatic cascades: a linguistically sound model of pronunciation by analogy. In: Proceedings of 35th ACL, pp. 429–435 (1997)
7. Lepage, Y., Shin-ichi, A.: Saussurian analogy: a theoretical account and its application. In: 7th COLING, pp. 717–722 (1996)
8. Yvon, F.: Finite-state machines solving analogies on words. Technical Report D008, École Nationale Supérieure des Télécommunications (2003)
9. Yvon, F., Stroppa, N., Delhay, A., Miclet, L.: Solving analogies on words. Technical Report D005, École Nationale Supérieure des Télécommuncations, Paris, France (2004)
10. Stroppa, N., Yvon, F.: Formal Models of Analogical Proportions. Available on HAL Portal (2007)
11. Miclet, L., Bayroudh, S., Delhay, A.: Analogical dissimilarity: definitions, algorithms and two experiments in machine learning. J. Artif. Intell. Res. **32**, 793–824 (2008)
12. Ben Hassena, A.: Apprentissage analogique par analogie de structures d'arbres. Ph.D. thesis, University de Rennes I, France (2011)
13. Bhargava, A., Kondrak, G.: How do you pronounce your name? improving g2p with transliterations. In: 49th ACL/HLT, Portland, USA, pp. 399–408 (2011)
14. Lavallée, J.F., Langlais, P.: Moranapho: un système multilingue d'analyse morphologique basé sur l'analogie formelle. TAL **52**, 17–44 (2011)
15. Kurimo, M., Virpioja, S., Turunen, V., Blackwood, G., Byrne, W.: Overview and results of morpho challenge. In: 10th Workshop of the Cross-Language Evaluation Forum (CLEF 2009). Lecture Notes in Computer Science, pp. 578–597 (2009)
16. Creutz, M., Lagus, K.: Inducing the morphological lexicon of a natural language from unannotated text. In: International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning (AKRR'05), Espoo, Finland, pp. 106–113 (2005)

17. Spiegler, S.: Emma: A novel evaluation metric for morphological analysis—experimental results in detail. Technical Report CSTR-10-004, University of Bristol, Bristol (2010)
18. Stroppa, N., Yvon, F.: An analogical learner for morphological analysis. In: 9th Conference on Computational Natural Language Learning (CoNLL), Ann Arbor, USA, pp. 120–127 (2005)
19. van den Bosch, A., Daelemans, W.: Data-oriented methods for grapheme-to-phoneme conversion. In: EACL, Utrecht, Netherlands, pp. 45–53 (1993)
20. Lepage, Y., Denoual, E.: Purest ever example-based machine translation: detailed presentation and assesment. Mach. Translat **19**, 25–252 (2005)
21. Lepage, Y., Lardilleux, A., Gosme, J.: The greyc translation memory for the iwslt 2009 evaluation campaign: one step beyond translation memory. In: 6th IWSLT, Tokyo, Japan, pp. 45–49 (2009)
22. Langlais, P., Patry, A.: Translating unknown words by analogical learning. In: EMNLP, Prague, Czech Republic, pp. 877–886 (2007)
23. Denoual, E.: Analogical translation of unknown words in a statistical machine translation framework. In: MT Summit XI, Copenhagen, Denmark, pp. 135–141 (2007)
24. Langlais, P., Yvon, F., Zweigenbaum, P.: Improvements in analogical learning: application to translating multi-terms of the medical domain. In: 12th EACL, Athens, pp. 487–495 (2009)
25. Koehn, P.: Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In: 6th AMTA, Washington DC (2004)
26. Somers, H., Sandapat, S., Naskar, S.K.: A review of ebmt using proportional analogies. In: 3rd Workshop on Example-Based Machine Translation, Dublin, Ireland, pp. 53–60 (2009)
27. Gosme, J., Lepage, Y.: Structure des trigrammes inconnus et lissage par analogie. In: 18e TALN, Montpellier, France (2011)
28. Claveau, V., L'Homme, M.C.: Structuring terminology by analogy-based machine learning. In: 7th International Conference on Terminology and Knowledge Engineering, Copenhagen, Denmark (2005)
29. Moreau, F., Claveau, V., Sébillot, P.: Automatic morphological query expansion using analogy-based machine learning. In: 29th European conference on IR research (ECIR'07), Berlin, Heidelberg, pp. 222–233 (2007)
30. Correa, W.F., Prade, H., Richard, G.: When intelligence is just a matter of copying. In: ECAI'12, pp. 276–281 (2012)
31. Langlais, P., Yvon, F.: Scaling up analogical learning. Technical report, Paritech, INFRES, IC2, Paris, France (2008)
32. Stroppa, N.: Définitions et caractérisations de modèles à base d'analogies pour l'apprentissage automatique des langues naturelles. Ph.D. thesis, Telecom Paris, ENST, Paris, France (2005)
33. Eisner, J.: Parameter estimation for probabilistic finite-state transducers. In: 40th ACL, Philadelphia, USA, pp. 1–8 (2002)
34. Lepage, Y., Lardilleux, A.: The greyc translation memory for the iwslt 2007 evaluation campaign. In: 4th IWSLT, Trento, Italy, pp. 49–54 (2008)
35. Ando, S., Lepage, Y.: Linguistic structure analysis by analogy: Its efficiency. In: NLPRS, Phuket, Thailand, pp. 401–406 (1997)
36. Dandapat, S., Morrissey, S., Naskar, S.K., Somers, H.: Mitigating problems in analogy-based ebmt with smt and vice versa: a case study with named entity transliteration. In: PACLIC, Sendai, Japan (2010)
37. Li, H., Kumaran, A., Pervouchine, V., Zhang, M.: Report of news 2009 machine transliteration shared task. In: Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration. NEWS '09, pp. 1–18 (2009)
38. Langlais, P.: Formal analogy for natural language processing: a review of issues to be adressed. In: 1st International Workshop Similarity and Analogy-based Methods in AI (ECAI Workshop), Montpellier, France, pp. 49–55 (2012)
39. Freund, Y., Schapire, R.E.: Large margin classification using the perceptron algorithm. Mach. Learn. **37**, 277–296 (1999)
40. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**, 273–297 (1995)

41. Eisner, J.: Parameter estimation for probabilistic finite-state transducers. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, pp. 1–8 (2002)
42. Lepage, Y.: Solving analogies on words: an algorithm. In: COLING-ACL, Montreal, Canada, pp. 728–733 (1998)