Issues in Analogical Learning over Sequences of Symbols: a Case Study with Named Entity Transliteration.

Philippe Langlais[1]

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 [18]. They can accommodate a wide variety of linguistic data without resorting to *ad hoc* representations [26] and are inherently good at capturing long dependencies between data. Unfortunately, applying analogical learning on top of formal analogy to nowadays large Natural Language Processing (NLP) tasks is very challenging. In this paper, we draw on previous works we conducted 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 $[x : y :: z : t]$, which reads as "x is to y as z is to t". While some works have been proposed for handling semantic relationships [32, 8], we focus in this study on formal proportional analogies (hereafter formal analogies or simply analogies), that is, proportional analogies involving relationships at the formal level, such as $[miracle : miraculeux :: fable : fabuleux]$.

Early work on formal analogies for NLP was devoted to propose computational definitions of proportional analogies. Yvon [34] proposed a definition where a prefixation or a suffixation operation was allowed between forms. In [22], 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 4 entities in (formal) analogical relation. Later on, Yvon et al. [36] proposed a model of analogy which generalizes the model of [22] 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 an letter to Mary* is to *Mary received an letter from Paul*). Stroppa [31] further extended this model to various algebraic structures, among which trees which are ubiquitous in NLP. Also, Miclet et al. [24] built on the definition of [36] 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 [2] .

Another thread of studies is devoted to applications of analogical learning to NLP tasks. Lepage [22] early proposed an analogical model of parsing which uses a treebank (a database of syntactically analyzed sentences). He conducted proof-of-concept experiments. Yvon [34] addressed the task of grapheme-to-phoneme conversion, a problem which continues to be studied thoroughly (e.g. [3]). In [17], the authors address the task of identifying morphologically related word forms in a lexicon, the main task of the MorphoChallenge evaluation campaign [13]. Their approach, which capitalizes on formal analogy to learn relations between words proved to be competitive with state-of-the-art approaches (e.g. [5]) and ranked first on the Finnish language according the EMMA metric (see [28]) which is now the official metric since Morphochallenge 2010. Stroppa and Yvon [31] applied analogical learning to computing morphosyntactic features to be associated with a form (lemma, part-of-speech, and additional features such as number, gender, case, tense, mood, etc.). The performance of the analogical device on the Dutch language was as good as or better than the one reported in [33].

Lepage and Denoual [19] 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 [14] investigated the more specific task of translating unknown words, a problem simultaneously investigated in [7]. In [16], 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 [12] trained at the character level, but the recall was lower. A simple combination of both systems outperformed significantly both engines. See [27] for a technical discussion of those works. Very recently, Gosme et Lepage [11] investigate the use of formal analogy for smoothing n-gram language models. They report improvements over fair baselines in different languages, but for small training corpora only.

Analogical learning has also been applied to various other purposes, among which terminology management [4], query expansion in Information Retrieval [25], classification of nominal and binary data, as well as handwritten character recognition [24]. 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 [27]. This motivates the present paper.

2 PRINCIPLE

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 of elements x_k. We call input set, and note it $\mathcal{I} = \bigcup_k i(x_k)$, 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

[1] University of Montreal, Canada, email: felipe@iro.umontreal.ca

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 define with t a proportional analogy 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;[2] 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 2-first steps of the inference as the *generator*, while we call the third one the *aggregator*.

Let's illustrate this on a tiny example where the task is to associate a sequence of part-of-speech (POS) tags to any given sentence considered 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 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[3] 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 source analogy is identified during step-1, or because the source analogies identified do not lead to analogies in the output space (failure of the inductive bias).

3 ISSUES WITH ANALOGICAL LEARNING

3.1 Formal Analogy

We mentioned that several definitions of formal analogy have been proposed. There are two of them that stand above the others in the sense that they can account for a larger variety of relations than the others: the one defined in [22] and the one defined in [36]; the latter generalizing the former. The choice of the definition to work with has some practical impact, since simpler relations (such as prefixation) are easier to recognize than more complex ones. Although we normally work with the second definition (because it is the most general one we know of), the discussion in this paper generally applies for all sensible definitions we know.

3.2 Searching in the Input Space

Identifying analogies in the input space (step-1) is a process cubic in the size of $\mathcal{I}$. Clearly, a brute-force approach would be manageable for toy problems only. This is why several authors have worked out some strategies we discuss in this section.

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

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

3.2.1 A quadratic search procedure

The search for input analogies can be transformed into a quadratic number of equation solving [19] 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 (a typical vocabulary in an NLP application has in the order of 10^5 words). Therefore, sampling has to be performed.

More precisely, for an element t to be treated, we solve analogical equations $[y : x :: i(t) :?]$ for some pairs $\langle x,y\rangle$ belonging to the neighborhood of $i(t)$. Those solutions that belong to the input space are the z-forms we are interested in. This strategy reduces the search procedure to the resolution of a number of analogical equations which grows quadratically with the size of the neighborhood set $\mathcal{N}$:

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

For instance, in [14] the authors deal with an input space in the order of tens of thousand forms by sampling x and y among the closest forms, in terms of edit-distance, to the form *i(t)*.

3.2.2 Exhaustive tree-count search

In [15], the authors developed algorithms for scaling up the search procedure. The main idea is to exploit a property of formal analogies [22]:

$$[x : y :: z : t] \Rightarrow |x|_c + |t|_c = |y|_c + |z|_c \quad \forall c \in \mathcal{A} \qquad (1)$$

where $\mathcal{A}$ 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 $\mathcal{C}(\langle x,t\rangle) = \{\langle y,z\rangle \in \mathcal{I}^2 \mid |x|_c + |t|_c = |y|_c + |z|_c \quad \forall c \in \mathcal{A}\}$ the set of pairs satisfying the count property with respect to $\langle x,t\rangle$.

Their strategy, called tree-count, 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[4], we collect a set of candidate triplets for t. A verification of those that actually define with t an analogy must then be carried out. Formally, they build:

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

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.[5]

3.2.3 Sampled tree-count search

The tree-count search strategy allows to *exhaustively* solve step 1 for reasonably large input spaces (tens of thousands of forms). However,

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

[5] Possibly involving filtering.

computing analogies in very large input space (hundreds of thousand of forms) remains computationally demanding, as the retrieval algorithm must be carried out $o(\mathcal{I})$ times. In this case, in [15], the authors proposed to sample the *x*-forms:

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

The authors proposed a sampling strategy which selects *x*-forms that share with *t* some sequences of symbols. To this end, input forms are represented in a k-dimensional vector space, whose dimensions are frequent symbol n-grams, where $n \in [\texttt{min};\texttt{max}]$[6]. A form is thus encoded as a binary vector of dimension `k`, in which the ith coefficient indicates whether the form contains an occurrence of the ith n-gram. At runtime, we select the `N` forms that are the closest to a given form *t*, according to a distance (i.e. cosine).

3.2.4 Checking for analogies

For all the aforementioned search strategies, we need to verify that 4 forms are indeed in analogical relation. Stroppa [29] proposed a dynamic programming algorithm for checking $[x : y :: z : t]$ when the definition in [36] is being 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 [15]:

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

where $s[\$]$ indicates the last symbol of s. A simple trick consists in calling for the verification of an analogy only for the quadruplets that pass this test.

3.2.5 Open issues

One can already go a long way with the sampled tree-count approach we described. Still, it is unclear which sampling strategy should be considered for a given application. The vector space model proposed in [15] seems to work well in practice, but more experiments should confirm this.

More fundamentally, none of the search procedures proposed so far take 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 [29], 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 working on this.

[6] Typical values are `min=max=3` and `k=20 000`.

3.3 Solving Equations

Algorithms for solving analogical equations have been proposed for both definitions of interest we mentioned. For the definition of [36], it can be shown [35] that the set of solutions to an analogical equation is a rational language, therefore we can build a finite-state machine for encoding those solutions. In practice however, the automaton is non deterministic, and in the worst case, enumerating the solutions can be exponential in the length of the sequences being involved in the equation. The solution proposed in [16] 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 samplings considered.[7] It is important to note that typically, a solver produces several solutions to an equation, many being simply spurious, which means that they obey the definition of formal analogy, but are not valid forms.

To illustrate this, Figure 1 reports the solutions produced to the equation $[even : usual :: unevenly :?]$ by our implementation of the solver defined in [16]. Clearly, many solutions are not valid forms in English, although they define proper solutions according to the definition of formal analogy proposed in [36]. Indeed, this definition recognizes no less than 72 different legitimate solutions, which we were able to produce with enough sampling ($\rho \geq 2000$) in less than a few tenth of milliseconds.

Figure 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.

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)

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 a special meaning here). Figure 2 reports the number of solutions produced as a function of 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 quite drastically. Clearly, enumerating all the solutions is not a good idea (too much 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 done entirely randomly. 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 [9].

[7] We leave this notion unspecified, read [16] for details.

Figure 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 _.

$\rho = 20$	$nb = 8$
$t = 0.0003$	$r = \phi$
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)	

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 rather 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, why our experiments indicate this is a important issue. In [20], Lepage and Lardilleux filter out solutions which contain sequences of symbols not seen in the output space of the training set. This typically leaves many solutions alive, including spurious ones. In [19], Lepage and Denoual propose to keep the most frequently generated solution. The rationale being that forms that are generated by various analogical equations are more likely to be good ones. Also, Ando and Lepage [1] show that the closeness of objects in analogical relations is another interesting feature for ranking solutions generated.

In [16], the authors investigate the use of a classifier trained in a supervised way to recognize good 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 $\mathcal{E}_o$ 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 $\mathcal{E}_o$, such as the frequency with which a solution is proposed. This means that it cannot be used to early filter the unlikely solutions generated.

Improving the classifier paradigm deserves further investigations. Notably, in [16], 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 for better understanding the limits of the approach.

As discussed in [1], it is intuitively more suited to see the problem of separating good 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.[8] Ranking the solutions proposed by the two-first steps of analogical learning must be investigated as a replacement of the classification solution proposed in [16].

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

3.5 Dealing with Silence

In most experiments we conducted, we faced the problem that the learning mechanism we described might produce no solution for a given entity. This might happen because no source analogy has been identified, or because the source analogies identified do not lead to target equations that have a 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 [16], 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 [6]. In [19], 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 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.

3.6 Learning over Tree Structures

Few authors have discussed the possibility of manipulating tree structures instead of sequences of symbols in analogical learning. Stroppa [29] proposed a definition of formal analogies 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 [36]. Based on this definition, the authors of [30] 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 [2] 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 [30], 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 [2] 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 [1] for a very similar setting.

4 CASE STUDY

4.1 Settings

In order to illustrate some of the elements we discussed in the previous section, we applied analogical learning to the task of transliterating English proper names into Chinese. The task we studied is part of the NEWS evaluation campaign conducted in 2009 [23]. Transliteration is generally defined as phonetic translation 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 [23]. Examples of transliteration from English proper names into Chinese are reported in Table 4.

The organizers of the NEWS campaign kindly provided us with the data that was distributed to the participants of the task. Its main characteristics are reported in Table 1, after the English letters have been lowercased. 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 (30 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 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 released 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 $\mathcal{L}_1$.

Table 1. Main characteristics of the English-Chinese data provided in NEWS 2009.

	$train$	dev	$test$
examples	31 961	2 896	2 896
EN symbols	26	26	26
CH symbols	370	275	283

examples	
Emission	埃米申
Blagrove	布格夫
Aposhian	阿 波 希 安

We ran two configurations of our *generator*: FULL-TC corresponds to the exhaustive tree-count setting described in Section 3.2.2, while SAMP-TC corresponds to the sampled version described in Section 3.2.3.[9] Since the number of source analogies identified can be quite large for some test forms, we enforced a timeout of 1 minute per English proper name for accomplishing step-1 of the inference in the FULL-TC setting and a timeout of 20 seconds for the SAMP-TC configuration. In both cases, the solver was run with a sampling rate of $\rho = 200$.

Regarding the classifier, we followed [16] and trained a voted-perceptron [10]. We computed a total of 19 features including the frequency of a solution, its rank in the list, input and output degrees (a notion defined for instance in [29]), language models likelihoods, etc. A greedy search over the feature set revealed that a handful of features only where useful. We trained the classifier over 5 000 epochs. The same classifier was used for both the FULL-TC and the SAMP-TC configurations we tested.

4.2 Monitoring Analogical Inference

We describe in the following the FULL-TC configuration, while Table 2 reports the figures of interest for both configurations. For the exhaustive configuration, the average time spent on step-1 per English form is 17 seconds. For 327 forms, the timeout applied, which means that we likely missed useful source analogies involving those forms. Most of the time spent during step-1 was devoted to check candidate analogies, that is, the quadruplets that pass the test in Equation 1. The trick we mentioned in Equation 2 avoided 63.8% of the verifications, a very nice speed up.

An average of 4 517 input analogies were identified per test form (with a maximum of 32 016); for 18 of them however, we could not identify any source analogy, leading to no response in those cases. Out of the 2878 test forms for which we could identify at least one source analogy, 2838 of them lead to an average of 487 output equations, the other 50 were left without answer. Solving all those equations led to an average of 405 solutions per test form (minimum 2, maximum 2221). Note that many equations solved did not lead to any 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 0.22 seconds (maximum 1.5s). In the end, we decided to keep up to the 100 most frequently generated solutions for a given test form (a solution is typically generated by several equations).

[9] The 1000-closest input forms to each English test forms where considered, based on a vector space representing the $k = 1000$ most frequent 3-grams of characters observed in $\mathcal{I}$, and the cosine distance.

Table 2. Main characteristics of the two configurations tested.

	FULL-TC	SAMP-TC
avg. time (step-1)	17s	2
avg. time (step-2)	0.22s	0.01s
number of timeouts	327	1
avg. input analogies	4517	158
avg. output equations	487	18
avg. number of solutions	405	37.5
silence (step-1)	18	76
silence (step-2)	50	249

It is interesting to note the discrepancy between the number of source analogies identified and the number of target equations effectively solved, which is much lower. This indicates either that the source analogies were in large part fortuitous, or that the inference bias (one analogy in the input space corresponds to an analogy in the output space) does not apply well for this task.

4.3 Evaluation

Table 3. Number of reference solutions among the 100-top frequent solutions proposed by the FULL-TC configuration. Read the text for more.

rank	nb	$r_{2374}\%$	$r_{all}\%$	nb	$r_{1659}\%$	$r_{all}\%$
1	1093	46.0	37.7	1410	85.0	48.7
2	1418	59.7	48.9	1627	98.1	56.2
3	1582	66.6	54.6	1657	99.9	57.2
4	1699	71.6	58.7	1659	100.0	57.3
5	1796	75.7	62.0	.	.	.
⋮	⋮	⋮	⋮	⋮	⋮	⋮
100	2374	100.0	82.0	1659	100.0	57.3

The left part of Table 3 reports the number of reference transliterations identified in the first $rank$ positions of the list of solutions proposed by the generator. We note that we could treat at most 2374 test forms correctly if we consider the 100-most frequently generated solutions produced. This represents only 82% of the test forms. Looking only at the most frequently generated solution[10], we observe that 37.7% of the test forms were transliterated correctly. This represents an accuracy of 46% (see r_{2374}) if we only consider the 2374 test forms where the reference transliteration was identified correctly among the first 100 solutions. These figures clearly show that being able to distinguish good from spurious solutions has the potential to improve the overall approach by more than 30 absolute points.

The right part of Table 3 indicates the performance of the FULL-TC inference after the aggregation step. Out of the 2374 test forms for which the correct solution was identified in the first 100 positions, only 1659 (70%) ones now receive a good solution. This shows that the classifier is too aggressive. On the other hand, 48.7% of the test forms now have the correct solution in the first position. This represents an increase of 11 absolute points over keeping the most-frequent solution. Actually, we can observe that for most of the test forms, the reference solution is either in the 2-first positions, either not present at all. Considering the fact that we did not spend much time for engineering features for the task, this is rather encouraging.

[10] Ties are broken randomly.

Table 4. Random excerpt of analogical transliterations produced by FULL-TC. r_c (resp. r) indicates the rank of the correct transliteration in the candidate list after (resp. before) the aggregation step. nb indicates the number of solutions generated. We replaced the Chinese characters we could not print correctly with our LaTeX processor by roman letters.

EN forms	reference	solutions	r_c	r	nb
auchter	x 克 特	x 克 特 (218)	1	1	380
sundell	森 德 y	森 德 *y* (692)	1	5	664
fannin	范 宁	范 妮 恩 (54)	ϕ	5	104
frere	弗 里 y	弗 里 *y* (6113)	1	1	630
shurkin	舒 金	舒 *y* 金 (237) 舒 金 (208)	2	3	386

Table 4 provides a random excerpt of the output produced by the FULL-TC configuration. Table 5 reports the results of our system as measured by the official metrics that were used to evaluate the different participating systems [23]. Clearly, our system is not among the leading ones. In fact, we would have ended up at the 14th rank according to accuracy (ACC); 18 systems participated to the 2009 exercise. Since our major goal was to monitor analogical learning, we did not put efforts yet 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 [6], 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 [6] 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-to-Indi transliteration task.

Table 5. Metrics used at the NEWS 2009 evaluation campaign. For comparisons, $1st$ and $last$ indicates respectively the first and last performing systems, as reported in [23].

metric	FULL-TC	SAMP-TC	$1st$	$last$
ACC:	0.486	0.308	0.731	0.199
F-score	0.772	0.612	0.895	0.606
MRR	0.527	0.330	0.812	0.229
MAP$_{ref}$	0.486	0.308	0.652	0.199

5 CONCLUSION

We presented a number of works on formal analogy dedicated to various NLP tasks. We discussed a number of issues that we feel remain to be investigated for the approach to meet higher acceptance among the NLP community. 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.

ACKNOWLEDGEMENTS

We thank the reviewers for their detailed and enriching comments.

REFERENCES

[1] Shinichi Ando and Yves Lepage, 'Linguistic structure analysis by analogy: Its efficiency', in *NLPRS*, pp. 401–406, Phuket, Thailand, (1997).
[2] Anouar Ben Hassena, *Apprentissage analogique par analogie de structures d'arbres*, Ph.D. dissertation, Univ. de Rennes I, France, 2011.
[3] Aditya Bhargava and Grzegorz Kondrak, 'How do you pronounce your name? improving g2p with transliterations', in *49th ACL/HLT*, pp. 399–408, Portland, USA, (2011).
[4] Vincent Claveau and Marie-Claude L'Homme, 'Structuring terminology by analogy-based machine learning', in *7th International Conference on Terminology and Knowledge Engineering*, Copenhagen, Denmark, (2005).
[5] M. Creutz and K. Lagus, 'Inducing the morphological lexicon of a natural language from unan- notated text', in *International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning (AKRR'05)*, pp. 106–113, Espoo, Finland, (2005).
[6] Sandipan Dandapat, Sara Morrissey, Sudip Kumar Naskar, and Harold Somers, 'Mitigating problems in analogy-based ebmt with smt and vice versa: a case study with named entity transliteration', in *PACLIC*, Sendai, Japan, (2010).
[7] Étienne Denoual, 'Analogical translation of unknown words in a statistical machine translation framework', in *MT Summit XI*, pp. 135–141, Copenhagen, Denmark, (2007).
[8] Nguyen Tuan Duc, Danushka Bollegala, and Mitsuru Ishizuka, 'Cross-language latent relational search: Mapping knowledge across languages', in *AAAI'11*, pp. 1237 – 1242, (2011).
[9] Jason Eisner, 'Parameter estimation for probabilistic finite-state transducers', in *40th ACL*, pp. 1–8, Philadelphia, USA, (2002).
[10] Y. Freund and R. E. Schapire, 'Large margin classification using the perceptron algorithm', *Mach. Learn.*, **37**(3), 277–296, (1999).
[11] Julien Gosme and Yves Lepage, 'Structure des trigrammes inconnus et lissage par analogie', in *18e TALN*, Montpellier, France, (2011).
[12] Philipp Koehn, 'Pharaoh: a beam search decoder for phrase-based statistical machine translation models', in *6th AMTA*, Washington DC, (2004).
[13] M. Kurimo, S. Virpioja, V.T. Turunen, G.W. Blackwood, and W.J. Byrne, 'Overview and results of morpho challenge 2009', in *10th Workshop of the Cross-Language Evaluation Forum (CLEF 2009)*, Lecture Notes in Computer Science, pp. 578–597, (2009).
[14] Philippe Langlais and Alexandre Patry, 'Translating Unknown Words by Analogical Learning', in *EMNLP*, pp. 877–886, Prague, Czech Republic, (2007).
[15] Philippe Langlais and François Yvon, 'Scaling up analogical learning', Technical report, Paritech, INFRES, IC2, Paris, France, (Oct. 2008).
[16] Philippe Langlais, François Yvon, and Pierre Zweigenbaum, 'Improvements in Analogical Learning: Application to Translating multi-Terms of the Medical Domain', in *12th EACL*, pp. 487–495, Athens, (2009).
[17] Jean-François Lavallée and Philippe Langlais, 'Moranapho: un système multilingue d'analyse morphologique basé sur l'analogie formelle', *TAL*, **52**(2), 17–44, (2011).
[18] Yves Lepage. De l'analogie rendant compte de la commutation en linguistique. Habilitation à diriger des recherches, Université Joseph Fourier, Grenoble I, France, 2003.
[19] Yves Lepage and Étienne Denoual, 'Purest ever example-based machine translation: Detailed presentation and assesment', *Mach. Translat*, **19**, 25–252, (2005).
[20] Yves Lepage and Adrien Lardilleux, 'The greyc translation memory for the iwslt 2007 evaluation campaign', in *4th IWSLT*, pp. 49–54, Trento, Italy, (2008).
[21] Yves Lepage, Adrien Lardilleux, and Julien Gosme, 'The greyc translation memory for the iwslt 2009 evaluation campaign: one step beyond translation memory', in *6th IWSLT*, pp. 45–49, Tokyo, Japan, (2009).
[22] Yves Lepage and Ando Shin-ichi, 'Saussurian analogy: A theoretical account and its application', in *7th COLING*, pp. 717–722, (1996).
[23] Haizhou Li, A. Kumaran, Vladimir Pervouchine, and Min Zhang, '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).
[24] Laurent Miclet, Sabri Bayroudh, and Arnaud Delhay, 'Analogical dissimilarity: Definitions, algorithms and two experiments in machine learning', *Journal of Artificial Intelligence Research*, 793–824, (2008).
[25] Fabienne Moreau, Vincent Claveau, and Pascale Sébillot, 'Automatic morphological query expansion using analogy-based machine learn-

ing', in *29th European conference on IR research (ECIR'07)*, pp. 222–233, Berlin, Heidelberg, (2007).
[26] Vito Pirrelli and François Yvon, 'The hidden dimesion: a paradigmatic view of data-driven nlp', *Journal of Experimental & Theoretical Artificial Intelligence*, **11**, 391–408, (1999).
[27] Harold Somers, Sandipan Sandapat, and Sudip Kumar Naskar, 'A review of ebmt using proportional analogies', in *3rd Workshop on Example-Based Machine Translation*, pp. 53–60, Dublin, Ireland, (2009).
[28] Sebastian Spiegler, 'Emma: A novel evaluation metric for morphological analysis - experimental results in detail', Technical Report CSTR-10-004, University of Bristol, Bristol, (2010).
[29] Nicolas Stroppa, *Définitions et caractérisations de modèles à base d'analogies pour l'apprentissage automatique des langues naturelles*, Ph.D. dissertation, Telecom Paris, ENST, Paris, France, 2005.
[30] Nicolas Stroppa and François Yvon, 'Formal Models of Analogical Proportions'. Available on HAL Portal, 2007.
[31] Nicolas Stroppa and François Yvon, 'An analogical learner for morphological analysis', in *9th Conf. on Computational Natural Language Learning (CoNLL)*, pp. 120–127, Ann Arbor, USA, (2005).
[32] P.D. Turney and M.L. Littman, 'Corpus-based learning of analogies and semantic relations', in *Machine Learning*, volume 60, pp. 251–278, (2005).
[33] Antal van den Bosch and Walter Daelemans, 'Data-oriented methods for grapheme-to-phoneme conversion', in *EACL*, pp. 45–53, Utrecht, Netherlands, (1993).
[34] François Yvon, 'Paradigmatic cascades: a linguistically sound model of pronunciation by analogy', in *In Proceedings of 35th ACL*, pp. 429–435, (1997).
[35] François Yvon, 'Finite-state machines solving analogies on words', Technical Report D008, École Nationale Supérieure des Télécommunications, (2003).
[36] François Yvon, Nicolas Stroppa, Arnaud Delhay, and Laurent Miclet, 'Solving analogies on words', Technical Report D005, École Nationale Supérieure des Télécommuncations, Paris, France, (2004).