Reducing Noise Sensitivity of Formal Analogical Reasoning Applied to Language Transfer

Vincent Letard[1,2], Gabriel Illouz[1,2], and Sophie Rosset[1]

[1] LIMSI, CNRS, Université Paris Saclay
firstname.lastname@limsi.fr

[2] Université de Paris-Sud, F-91405 Orsay Cedex, France

Abstract. Previous applications of formal analogical reasoning to the task of natural to formal language transfer have shown a very high precision. However, this also came with a low recall. Analogical reasoning methods are very sensitive to noise in the input data. In order to improve the response rate, we propose to release the constraints. Firstly, we present a relaxation of the search in the example base. Secondly, we adapt a dynamic programming resolution algorithm from an existing work. We then discuss the genericity obtained. The results show the expected increase in recall. In particular, the first relaxation shows an interesting improvement of the correct answers.

1 Introduction

The design of interactive assistants is a nowadays trend as more and more complex procedures are operated using computers. In this work, we consider the applicative goal is of giving instructions to the computer in natural language, as shown in the example below for English and `bash`.

"Print 14 copies of ex08.pdf" → `lp -n 14 ex08.pdf`

This task is meant for an interactive application, where the user needs to perform operations but does not know or cannot remember the right command.

This paper focuses on transferring user specific language, or idiolects, into formal language. The experimental context is seen as a task of machine translation: natural to formal language transfer.

Example-based machine translation (EBMT) is a good candidate to address this task, as it adapts well to new, specific domains with few examples. Among them, formal analogical reasoning (FAR) operates directly on mere surface forms. The precision obtained is also quite high [8]. Analogy is known to be a crucial mechanism to human cognition [3]. However, the framework of FAR has limitations. Firstly, the complexity of the exhaustive computation is high. Some heuristics and optimization have been proposed. Secondly, the constraints on the input examples are strong. This makes formal analogy very sensitive to noise. The system remains silent when the input does not perfectly form an analogy with the examples from the base.

In Proceedings of the ICCBR 2016 Workshops. Atlanta, Georgia, United States of America

We propose two approximations to reduce noise sensitivity. The first one helps to consider more examples from the base even if they do not form an analogy *per se*. The second one allows approximating the process of resolution. Our hypothesis is that relaxing the strong constraints on FAR increases the recall and hopefully gives new, meaningful answers.

In the next section, we give some general context related to EBMT and FAR. The latter is introduced in more details in section 3. Our contributions are detailed in sections 4 and 5, after introducing their contexts. Section 6 describes the associated experiments. Finally, section 7 is dedicated to interpreting the results in comparison with related work.

2 Background

Example-based approaches to machine translation received an increasing interest since Nagao's paper [12]. EBMT and translation by analogy have been used as synonyms until the definition of formal models for analogical proportions [7, 14, 15]. This model is associated with a framework of two main algorithms, one for the resolution and the other for the validation of analogies. They are used to build the answer using a triplet of sentences. The task of natural (NL) to formal language (FL) transfer has already been tackled in [5], and [1, 2, 4] consider the mapping of instructions directly to an action model.

FAR algorithms are computationally demanding. Efforts have been made in the direction of reducing the computation time for the phase of searching in the example base. In order to prevent slowing down on big example bases, several heuristics have been used to guide and prune the search [8, 9]. A review of the heuristics for FAR is given in [13]. Langlais and Yvon [6] proposed an optimization based on the structure of count tree. It relies on the preservation of analogical proportions through morphism. We based our work on this optimization, more details are given in section 4.

Regarding the issue noise sensitivity, [11] proposed a general algorithm for solving formal analogical equations with the best approximation, according to a notion of analogical dissimilarity. It is presented in section 5.1.

[10] already applied FAR to the task of natural to formal language transfer. This work is used here as a baseline to assess our experimental results.

3 Formal Analogy for Language Transfer

3.1 Analogical Proportions on Sequences

A proportional analogy is a relation between 4 elements that can be stated in natural language as the expression "X is to Y as Z is to T". As an illustration, the following quadruplet is an analogical proportion over the domain of word sequences: "*Display f.txt* is to `cat f.txt` as *Display file* is to `cat file`". We use the notation $[X : Y :: Z : T]$, for a valid proportion, and $[X : Y :: Z : ?]$ for an analogical equation. These objects are processed using the operations of

verification of a proportion, and **resolution** of an equation. When applied to sequences, these operations rely on the identification of an alignment of the symbols. Each atomic alignment of symbols must be of one of the following alternating forms[1]: $(a,\ b,\ a,\ b)$ or $(a,\ a,\ b,\ b)$. A proportion is verified (valid) if it can be obtained from a concatenation of proportional atomic alignments. A sequence is a solution of an equation if the resulting proportion is verified. The solution to an analogical equation is often not unique, therefore the solution is selected from the shortest alignment as it is considered [14, 13] to produce a satisfying answer.

The exponential number of n to n alignments between sequences makes the naive algorithms not usable in real conditions. Efficient algorithms for performing the operations of validation and resolution are described in [14, 15]. Their complexity is in $\mathcal{O}(n^3)$, n being the size of the longest sequence.

Typically, the solving algorithms work by filling a three dimensional matrix of size $|X| \times |Y| \times |Z|$ where $[X\ :\ Y\ ::\ Z\ :\ ?]$ is the equation to be solved. The search in the matrix corresponds to the concurrent and ordered iteration over the symbols of the three sequences with the following legal transitions:

- reading identical symbols on X and Y
- reading identical symbols on X and Z
- reading a symbol on Y and outputting it
- reading a symbol on Z and outputting it

Note that the symbols can be characters, words, or any kind of predefined segment. The details of the algorithms are given using the term "symbol" to emphasize on their genericity.

3.2 Language Transfer

FAR can be applied to the problem of language transfer by processing sentences as sequences of symbols. In our case, the sentences are NL requests and FL commands, and the symbols are words. Here, we call *command* (c_i) the string of characters that is typed by the user in the terminal (*e.g.* "`ls -l folder/`"). Similarly, we call *request* (r_i) the whole string of characters that is submitted by the user. The example base B is a set of paired sentences. The request domain R is French[2] and the command domain C is `bash`. $B\ \subset\ (R\ \times\ C)$ Given a request $r \in R$, the base is searched for analogies in order to generate an answer s. We can search directly for equations $[r_i : c_i :: r : ?]$ or indirectly, for proportions $[r_i : r_j :: r_k : r]$ in order to solve the associated equation $[c_i : c_j :: c_k : ?]$.

These strategies are illustrated in examples 1.1 and 1.2. They are referred to as the **direct** and the **indirect** strategies respectively. The first example gives the solution "`lp -n 2 file.pdf`", obtained by mixing languages, while the second one produces "`gcc f.c`" only using the `bash` part of known forms. Both strategies allow the production of relevant, unseen commands, and are also complementary [10].

[1] a or b can be the empty symbol ϵ.
[2] The examples are given in English for convenience.

(r_i) *Print 14 copies of ex08.pdf* : (c_i) `lp -n 14 ex08.pdf`
::
(r) *Print 2 copies of file.pdf* : `?`

Example 1.1: Direct analogical resolution

(r_i) *Count the lines in f.c* : (r_j) *Count the lines in the C file f*
::
(r_k) *Compile f.c* : (r) *Compile the C file f*

(c_i) `wc -l f.c` : (c_j) `wc -l f.c`
::
(c_k) `gcc f.c` : `?`

Example 1.2: Indirect analogical resolution

In the next two sections we present approaches for optimizing search and relaxing the analogical resolution.

4 Exploring the Example Base

This section describes the tree-count search algorithm, proposed by Langlais and Yvon [6], and our adaptation for relaxing its associated constraints.

4.1 Tree-Count Search

The algorithm relies on the following property of formal analogy:

$$[X : Y :: Z : T] \Rightarrow |X| + |T| = |Y| + |Z|$$

That is, the count of symbols in X and T equals that in Y and Z. This also stands for any specific symbol of the lexicon $\mathcal{L}$ ($\mathcal{L}$ is the set of all the symbols appearing in B).

$$[X : Y :: Z : T] \Rightarrow \forall s \in \mathcal{L}\ |X|_s + |T|_s = |Y|_s + |Z|_s$$

The tree-count is built by assigning each form to be indexed to the path in the tree corresponding to the count vector of its symbols. An ordering is set on the lexicon so that each depth level corresponds to a token.

The naive search algorithm for finding proportions, given a user request r_u, works by trying every possible triplet of requests from the base and test them using the `validation` procedure. Using the tree-count, only a linear search of the base is needed. For each request r_1 of the base, the count vector of $r_1 \cup r_u$ is computed, that is the values of $|r_1|_s + |r_u|_s$ for every symbol s. Given this vector, the tree is searched for pairs (r_2, r_3) of forms that fit the count equation, as given previously. The complexity of the search is then $\mathcal{O}(|B| \times |\mathcal{L}|)$. While this reduces the computation time, the issue of sensitivity to the noise in the base or in the input remains.

4.2 Relaxed Tree-Count Search

In order to enable approximations at the step of triplet searching for indirect resolution, we adapted the tree-count algorithm to be more flexible to noise. Instead of looking for quadruplet of requests (3 from the base, 1 from the user) that are analogical proportions, our approach also explores some that are not. However, one does not want to consider each of the $|B|^3$ quadruplets that can be formed. We want to keep track of the distance from which a given quadruplet is to be an analogical proportion. We define the **deviation** Δ of a quadruplet of sequences (X, Y, Z, T) as follows[3]:

$$\Delta = \sum_{s \in \mathcal{L}} abs\left((|X|_s + |T|_s) - (|Y|_s + |Z|_s)\right)$$

The exact tree-count search algorithm retains from a step i only the paths that verify the counts for all symbols before i. With the relaxed algorithm, only the paths for which the counts are verified with a deviation Δ are selected for the next step. Algorithm 1 achieves a generic tree-count search for a given Δ. It was designed on the basis of the standard tree-count search from Langlais and Yvon [6]. The tree-count is searched breath-first. Each pair of nodes satisfying the constraints on the count of symbols is added to the list named `frontier`. The paths from other pairs are pruned. We keep track of the current δ while descending the tree, thus each step is performed $\Delta - \delta$ times. The macro $eqSumChildren(n_1, n_2, \delta, adj)$ retrieves every pair of children (c_1, c_2) – from n_1 and one from n_2 – for which the difference between the goal count and the sum of the occurrences of the current symbol i is less than adj.

Algorithm 1: Relaxed search using a count-tree

Input: A pair of forms $P_{in} = (f^1_{in}, f^2_{in})$, a maximum divergence Δ and a count-tree T using the lexicon $\mathcal{L}$
Output: The set of pairs $P_{out} = (f^1_{out}, f^2_{out}, \delta)$ from T such that $\delta = |count(P_{in}) - count(P_{out})| \leq \Delta$

$counts \leftarrow encode(s^1_{in}, s^2_{in})$
$frontier \leftarrow \{(root(T), root(T), 0)\}$
$i \leftarrow 1$
while $i \leq |\mathcal{L}| \wedge frontier \neq \{\}$ **do**
  $res \leftarrow \{\}$
  $\nu \leftarrow counts[\mathcal{L}[i]]$
  forall the $(n_1, n_2, \delta) \in frontier$ **do**
    for $adj = 0 \rightarrow \Delta - \delta$ **do**
      $res \leftarrow res \cup eqSumChildren(n_1, n_2, \delta, adj)$
    end
  end
  $frontier \leftarrow res$
  $i \leftarrow i + 1$
end
return $\{(f_1, f_2, \delta) : f_1 \in p_1.forms, f_2 \in p2.forms, (p_1, p_2, \delta) \in frontier\}$

The proposed relaxation of the algorithm has the effect of retaining up to $2\Delta+1$ times more open paths at each step. The coefficient 2 comes from that the

[3] $abs()$ denotes the absolute value for distinction from the count of symbols.

deviation can be about adding or removing a symbol. The worst case complexity however remains in $\mathcal{O}(n^3)$ with n the size of the base, since no more than every triplet can be explored. This is of course limited by the actual number of paths in the tree, that is, the set of count-compatible requests pairs (n^2). This set is expected to be much smaller than the n^2 for a given input request pair, especially for long sentences. However, it also depends on Δ and on the dispersion of the examples in the base.

The relaxed tree-count search helps decreasing noise sensitivity of formal analogy on the source, *i.e.* on requests. However, it has no effect on the noise in the target domain, *i.e.* commands. The following section tackles this issue with a relaxation of the analogical resolution algorithm.

5 Approximate Resolution

5.1 Analogical Dissimilarity

The notion of analogical dissimilarity (AD) introduced by Miclet *et al.* [11] intends to represent how far a quadruplet of objects is to be a valid analogical proportion. Hence, an AD of 0 signifies that the quadruplet is a valid proportion. Analogical dissimilarity is defined for quadruplets of sequences as the alignment of the symbols of minimal cost. The cost of an alignment itself is defined as the sum of the AD of each quadruplet of symbols. Finally, the AD of a quadruplet of symbols can be set depending on the symbols themselves. In [11], it is assumed that a matrix of similarity is given for the whole alphabet. This is not realistic if the symbols are words. The general AD between symbols can be formally defined similarly as for binary values: the AD between a quadruplet of symbols is the minimum number of symbols that have to be substituted in order for the quadruplet to stand as a valid analogical proportion. For example $AD(a, b, c, b) = 1$. This definition bounds the AD for symbols to values between 0 and 2. Hence, the AD for sequences is comprised between 0 and $2n$, n being the size of the alignment of minimal cost. The example below shows an alignment (of minimal cost) of 4 sequences, along with the costs of each symbol quadruplet alignment. The total cost of the quadruplet of sequences is $AD(X, Y, Z, T) = 4$.

X	*remove*	ϵ	*the*	*file*	ϵ	*f.zip*
Y	*show*	ϵ	*the*	*file*	*named*	*foo.txt*
Z	*delete*	ϵ	*the*	*file*	ϵ	*f.sh*
T	*show*	*me*	*the*	*file*	*named*	*bar.txt*
	1	*1*	*0*	*0*	*0*	*2*

Note that [11] only consider character level segmentation, the example is segmented at word level in order to represent the use of AD in our context.

The authors then propose an algorithm for computing approximate solutions to analogical equation using AD. The solving process (SOLVANA) is based on the dynamic programming analogical resolution algorithm [14]. At each step (i, j, k) of the filling of the three dimensional matrix, every previous cells are examined, that is all the cells of indices $(i - d_i, j - d_j, k - d_k)$ for all $(d_i, d_j, d_k) \in \{0, 1\}^3 \setminus (0, 0, 0)$. The path that corresponds to the alignment of the lowest AD

is selected. This is determined by enumerating every character of the alphabet and computing the AD of the corresponding quadruplet of symbols.

The complexity of the algorithm is $\mathcal{O}(m * n^3)$ where m is the size of the lexicon, and n is the maximum length of the sequences. In our context, the lexicon is a lot bigger than the set of all the characters used by Miclet. Moreover, unlike the set of the characters, a lexicon of words can never be exhaustive, and is always subject to updates. Hence we cannot use the algorithm as is.

5.2 Analogical Dissimilarity on Arbitrary Sized Lexicons

As for the SOLVANA algorithm [11], our proposition is based on the dynamic programming solving algorithm. However, less transitions are considered for each cell. In the cell (i, j, k), the possible transitions are from the cells:

	X	Y	Z
$t1$:	$i-1$,	$j-1$,	k
$t2$:	$i-1$,	j ,	$k-1$
$t3$:	i ,	$j-1$,	k
$t4$:	i ,	j ,	$k-1$
$t5$:	$i-1$,	j ,	k

This choice was guided by the fact that every other transition can be reduced to a combination of these ones. Among them, the legal transitions are:

1. read identical symbols on the X and Y sequences, write nothing on T
2. read identical symbols on the X and Z
3. read a symbol from Y and write it to T
4. read a symbol from Z and write it to T

They induce no deviation, as they are legitimate for a valid analogy. Below are the five selected operations that are not legitimate.

1. read different symbols on X and Y, no output (virtual substitution)
2. similarly, read different symbols on X and Z
3. read a symbol from Y, write nothing on output (virtual deletion)
4. similarly, for Z
5. similarly, for X

Each illegitimate transition can be used at a cost of 1. As in 4.2, we name this cost **deviation** – distinguishing from dissimilarity. Consequently, the four transitions listed previously have a deviation of 0.

Substitution and deletion are considered but inserting new symbols implies to get them from a source. However, enumerating every possible word to insert from the lexicon is too costly. Also, identifying relevant insertions would require a non trivial strategy for guiding the search. Hence, we did not consider the insertion of new symbols.

As for the algorithm of relaxed search, the relaxed resolution takes the deviation threshold as parameter. During the filling of the three dimensional matrix, deviations are considered up to the specified threshold. This makes the complexity of the algorithm in $\mathcal{O}(n^3)$, that is equivalent to the initial algorithm.

6 Experiments and Results

6.1 Experimental Setup

Our setup is based on the setting and corpus used in [10]. It is composed of 421 associations between requests in French and commands in `bash`. We used the dynamic programming algorithm from [14] for the resolution.

The system takes a natural language request as input and the output is a command. The direct and indirect resolution strategies are performed independently. The direct resolution module searches the base for mixed triplets and then solves them with the minimum possible deviation $\delta \leq \Delta_r$. The indirect resolution module searches the example base for all the quadruplets of requests of the minimum possible deviation $\delta \leq \Delta_s$ using the tree-count index. The commands associated with the triplets found are then passed to the resolution step which attempts to solve the resulting equation with the minimum $\delta \leq \Delta_r$.

We used two data sets for our experiments[4]. The first one, called SP (Same Person), is composed of associations collected with the same person who wrote the examples of the base, and the commands still used the same set of parameters. The second one, called NP (New Person), contains associations written by another person while systematically changing the parameters of each command. Both test sets contain 77 requests and their associated commands for evaluation.

6.2 Results

The tests were conducted by varying the allowed deviation using both sets to assess the score. Table 1a gives the scores and execution time for the SP and NP test sets. The lines correspond in that order to the number of answers given, the number of correct answers, the associated values of precision and recall, the average time per request for an exhaustive search, and the average time per request when ending the search immediately after the first result found. For the SP set, the number of answers is consistently increasing with the allowed deviation, up to 64 of 77. Similarly on the NP set, though with a lower increase (35 of 77). There are less new correct answers on NP than on SP. Besides, the execution time grows a lot as soon as $\Delta_s > 2$, especially with the SP set.

The tests with approximate resolution only are far less time consuming (*cf.* table 1b). However, despite an important increase in recall, the precision drops more than with the relaxed search only. The number of correct answers hardly increases by few units compared to the default run ($\Delta_r = 0$) and stagnates for $\Delta_r \geq 2$. These results were partly expected, but show interesting specificities, which are discussed in the next section.

[4] The datasets can be found at http://perso.limsi.fr/letard/ilar/iccbr_analogy2016.zip

Table 1: Test results

(a) Increasing the search deviation

Evaluating on SP						
Δ_s	0	1	2	3	4	5
answers	36	43	50	53	59	64
correct	35	42	49	51	57	61
precision	0.97	0.98	0.98	0.96	0.97	0.95
recall	0.47	0.56	0.65	0.69	0.77	0.83
time/req*	0.4	2.4	14.1	35.5	82.1	163.8
time/req	0.3	1.3	5.0	13.7	32.9	67.0
Evaluating on NP						
Δ_s	0	1	2	3	4	5
answers	9	10	14	18	27	35
correct	9	10	10	11	11	11
precision	1	1	0.71	0.61	0.41	0.31
recall	0.12	0.13	0.18	0.23	0.35	0.45
time/req*	0.0	0.5	3.1	10.5	27.0	56.4
time/req	0.0	0.3	2.2	8.1	20.2	42.1

(b) Increasing the solving deviation

Evaluating on SP						
Δ_r	0	1	2	3	4	5
answers	36	48	55	61	66	71
correct	35	41	42	42	42	42
precision	0.97	0.85	0.76	0.69	0.63	0.59
recall	0.47	0.62	0.71	0.79	0.85	0.92
time/req*	0.4	0.9	1.3	1.8	2.2	2.7
time/req	0.3	0.4	0.5	0.5	0.6	0.6
Evaluating on NP						
Δ_r	0	1	2	3	4	5
answers	9	20	30	39	44	52
correct	9	10	12	12	12	12
precision	1	0.50	0.40	0.31	0.27	0.23
recall	0.12	0.26	0.39	0.51	0.57	0.68
time/req*	0.4	0.9	1.3	1.8	2.2	2.7
time/req	0.3	0.4	0.5	0.5	0.6	0.6

7 Contrastive Discussion

7.1 Findings

The relaxation of the counting constraint for searching requests in the example base allowed a remarkable increase of the recall of the system, with a minimal impact on its precision on the SP set. The most limiting factor for an application is actually the execution time which rapidly grows out of control. Also, the time complexity has to be watched carefully while increasing the size of the example base. Efficient selection strategies may be necessary in order to continue using the counting relaxation within reasonable delays. The progress are far lower while testing on the NP test set. Indeed, as was highlighted in [10], successful analogical resolution on this particular test set mostly involves direct analogies. The counting relaxation only affects the indirect resolution, as counting is only used at the quadruplet searching phase.

To summarize, while the relaxation allows relevant approximations in the source language quadruplet, the target language triplet still has to have an exact analogical solution in order to produce the answer. This also applies to mixed triplets for direct resolution. Although we must be cautious while making strong conclusions due to the small size of our example base, it can be conclusively said that FAR has an important progression margin towards noise handling. As for the results on our corpora, applying a search deviation of 2 words seems to be a good compromise between computing time, precision and recall for a practical application.

7.2 Related Work

Our modification of the dynamic programming analogical resolution algorithm, serves a different purpose than the one proposed by [11]. Their use is focused on the production of outputs containing deviations with the concern of giving an answer as often as possible when the deviation is not too high. However, enumerating the set of all the tokens in order to consider every possible insertion, even while eliminating most of them, cannot be done while segmenting at word level. The value of analogical dissimilarity is defined between characters by a predefined set of features on the alphabet. This is an interesting point to explore in our case, in order to weight more accurately the approximate substitutions of words. It may also be automatically determined using some similarity on words.

8 Conclusion and Future Work

We presented two propositions of relaxation on two algorithms used for language transfer by analogy. Our hypothesis was that these relaxations could improve the recall with a limited impact on the precision.While the approximate analogical resolution did not yield very high scores, the relaxed tree-count search, despite a high computational cost, showed a promising increase of the correct answers.

Since our application falls whithin human-computer interaction, one way the system may be improved is by asking a confirmation to the user before choosing the answer. Indeed, after performing a relaxed tree-count search, the validation step is not relevant, as the deviation is higher than 0. However, three of the four sentences can be set as an analogical equation and solved. This gives an alternative version of the fourth sentence. This version cannot be used to create an answer because it is absent from the example base, however it can be useful in interactive context. If no result is found with 0 deviation, the confidence in the output is supposed to decrease. Also, possibly many solutions with deviation can be returned, not all of them being correct. In this case, the system can use the generated request in order to ask the user for a pre-confirmation. If positive, it will higher the confidence in the corresponding solution. If negative, the system continues with the next solution.

This work suggests other interesting perspectives on several aspects of the approaches. An important track is open towards reducing the computation time of the relaxed tree-count search. As we showed, greater values of deviation keep producing new correct answers with a limited noise in output, but the delay becomes unrealistic. One may attempt to reduce it with a more informed definition of deviation that depends on the content of the tokens, or possibly on some external knowledge. Another track is to change the token unit to a sequence of words where this is possible, this would enable the easier processing of deviation between multiword expressions. Types of deviations may also be considered in order to avoid the deletion or replacement of crucial tokens such as parameters.

References

1. Branavan, S., Chen, H., Zettlemoyer, L.S., Barzilay, R.: Reinforcement learning for mapping instructions to actions. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1. pp. 82–90. Association for Computational Linguistics (2009)
2. Branavan, S., Zettlemoyer, L.S., Barzilay, R.: Reading between the lines: Learning to map high-level instructions to commands. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. pp. 1268–1277. Association for Computational Linguistics (2010)
3. Hofstadter, D.R., Sander, E.: Surfaces and Essences. Basic Books (2013)
4. Kiddon, C., Ponnuraj, G.T., Zettlemoyer, L., Choi, Y.: Mise en place: Unsupervised interpretation of instructional recipes. Conference of Empirical Methods in Natural Language Processing (2015)
5. Kushman, N., Barzilay, R.: Using semantic unification to generate regular expressions from natural language. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics. North American Chapter of the Association for Computational Linguistics (NAACL) (2013)
6. Langlais, P., Yvon, F.: Scaling up analogical learning. In: COLING (Posters). pp. 51–54 (2008)
7. Lepage, Y.: Solving analogies on words: An algorithm. In: Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1. pp. 728–734. ACL '98, Association for Computational Linguistics, Stroudsburg, PA, USA (1998), http://dx.doi.org/10.3115/980845.980967
8. Lepage, Y., Denoual, E.: Purest ever example-based machine translation: Detailed presentation and assessment. Machine Translation 19(3-4), 251–282 (2005)
9. Lepage, Y., Lardilleux, A.: The greyc machine translation system for the iwslt 2007 evaluation campaign. In: IWSLT 2007. pp. 49–54 (2007)
10. Letard, V., Illouz, G., Rosset, S.: Analogical reasoning for natural to formal language transfer. In: International Conference on Tools with Artificial Intelligence (2015)
11. Miclet, L., Bayoudh, S., Delhay, A.: Analogical dissimilarity: definition, algorithms and two experiments in machine learning. Journal of Artificial Intelligence Research pp. 793–824 (2008)
12. Nagao, M.: A framework of a mechanical translation between japanese and english by analogy principle. Artificial and human intelligence pp. 351–354 (1984)
13. Somers, H., Dandapat, S., Naskar, S.K.: A review of ebmt using proportional analogies. EBMT 2009 - 3rd Workshop on Example-Based Machine Translation (2009)
14. Stroppa, N.: Analogy-Based Models for Natural Language Learning. Phd thesis, Télécom ParisTech (Nov 2005), https://tel.archives-ouvertes.fr/tel-00145147
15. Stroppa, N., Yvon, F.: Formal models of analogical proportions. Tech. rep., École Nationale Supérieure des Télécommunications, Paris, France (2007)