Character–position arithmetic for analogy questions between word forms

Yves Lepage*

IPS, Waseda University
2-7 Hibikino, Wakamatsu-ku, Kitakyushu-shi, 808-0135 Fukuoka-ken, Japan
yves.lepage@waseda.jp

Abstract. We show how to answer analogy questions $A : B :: C : D$ of unknown D between word forms, by essentially relying on the basic arithmetic equality $D[i_B - i_A + i_C] = B[i_B] - A[i_A] + C[i_C]$ on characters and positions at the same time. We decompose the problem into two steps: specification and decoding. We examine several techniques to implement each of these two steps. We perform experiments on a set of positive and negative examples and assess the accuracy of combinations of techniques. We then evaluate the performance of the best combination of techniques on a large set of more than 40 million analogy questions from the training data of a shared task in morphology. We obtain the correct answer in 94 % of the cases.

Keywords: Formal analogy, analogy questions, character–position arithmetic.

1 Introduction

In this paper, we address the problem of answering analogy questions of the type $A : B :: C : D$ between word forms where the unknown is D. Our proposal consists in relying essentially on the intuitive basic arithmetic equality $D = B - A + C$. We propose to write this arithmetic equality using characters and positions at the same time:

$$D[i_B - i_A + i_C] = B[i_B] - A[i_A] + C[i_C] \tag{1}$$

The use of this arithmetic equality is directly inspired by the famous equality between vectors proposed in [7] to answer analogy questions between words in the framework of distributional semantics. This is now referred to as vector arithmetic and is always exemplified with:

$$\overrightarrow{\text{queen}} \approx \overrightarrow{\text{king}} - \overrightarrow{\text{man}} + \overrightarrow{\text{woman}} \qquad \text{(man : king :: woman : queen)} \tag{2}$$

* This work was supported by a JSPS Grant, Number 15K00317 (Kakenhi C), entitled Language productivity: efficient extraction of productive analogical clusters and their evaluation using statistical machine translation.

This paper gives empirical support for the use of Equality (1) to answer analogy questions (which only involve commutation[1]) between word forms (not meaning). For relevance to morphology, no knowledge other than equality of characters is used, i.e., order of letters in an alphabet, or the like, is not used.

The rest of the paper is structured as follows. Section 2 shows how the answer to an analogy question can be specified by a character–position matrix computed using Equality (1). Specification refinements are introduced to solve problematic cases. Section 2 shows that decoding the answer from the character–position matrix can be viewed as an assignment problem and thus solved using a standard algorithm. Section 4 summarises the two previous sections by giving an algorithm for the proposed method. This proposed method is validated by two series of experiments in Section 5.

2 Specifying the answer of an analogy question

2.1 Known features of the answer to an analogy question

From previous research [4,10,2,3], it is known that the answer D to $A : B :: C : D$ is partially determined. In particular, its length, which characters it contains, and their number of occurrences, are known. In mathematical notations:

$$A : B :: C : D \quad \Rightarrow \quad \begin{cases} |D| = |B| - |A| + |C| \\ |D|_c = |B|_c - |A|_c + |C|_c, \ \forall c \end{cases} \tag{3}$$

where $|X|$ stands for the length of string X and $|X|_c$ for the number of occurrences of character c in string X. The above equations are yet another instance of the general arithmetic equality $D = B - A + C$. In addition, some work [6] states that the LCS distance, noted d below, between the pair of terms is equal:

$$A : B :: C : D \quad \Rightarrow \quad \begin{cases} d(A, B) = d(C, D) \\ d(A, C) = d(B, D) \end{cases} \tag{4}$$

2.2 Character–position arithmetic

In analogies of commutation, pieces are combined and exchanged to compose the four different terms of the analogy A, B, C and D. It is always possible to put some pieces in common to A and B in correspondence with some pieces in common to C and D (or the same by exchanging B and C in this statement). For instance, in

$$\text{hearty} : \text{unheartily} :: \text{lucky} : \text{unluckily}, \tag{5}$$

[1] See [5] or [6, middle of p. 161]. We exclude analogies of repetition, e.g., Indonesian *guru* : *guru-guru* :: *pelajar* : *pelajar-pelajar*; reduplication, e.g., Ancient Greek λύω : λέλυκα :: παύω : πέπαυκα; and mirroring, e.g., *abc* : *wxyz* :: *cba* : *zyxw*.

the pieces 'heart', common to A and B, and 'luck', common to C and D, correspond. They are exchanged on each side of the symbol ::. We can write $A[1\ldots5] = B[3\ldots7] = \text{heart}$ and $C[1\ldots4] = D[3\ldots6] = \text{luck}$; and further

$$D[3\ldots6] \;=\; \text{luck} \;=\; B[3\ldots7] - A[1\ldots5] + C[1\ldots4] \;=\; \text{heart} - \text{heart} + \text{luck},$$

where $3 = 3-1+1$ and $6 = 7-5+4$. This equality combines several instances of Equation (1) for several instances of indices (i_A, i_B, i_C, i_D): $(1,3,1,3)$, $(2,3,2,3)$, $\ldots$, $(5,7,4,6)$

To summarise the previous remarks, Proposition (6), which embeds Equality (1), can be laid as a hypothesis to be tested. Importantly, it only makes sense if either $A[i_A] = B[i_B]$ or $A[i_A] = C[i_C]$ holds, i.e., if either (i_A, i_B) or (i_A, i_C) are *match points* in the matrices representing $A : B$ and $A : C$ (see Figure 1).

$$A : B :: C : D \;\Rightarrow\; \forall i_D \in \mathbb{N} \;/\; 1 \leq i_D \leq |D|, \quad \exists (i_A, i_B, i_C) \in \mathbb{N}^3 \;/ \quad \begin{cases} 1 \leq i_A \leq |A| \\ 1 \leq i_B \leq |B| \\ 1 \leq i_C \leq |C| \\ i_D = i_B - i_A + i_C \\ D[i_D] = B[i_B] - A[i_A] + C[i_C] \end{cases} \tag{6}$$

2.3 Specification of the answer using a character–position matrix

Enforcing Proposition (6) allows to specify the solution of an analogy question as illustrated in Figure 1. For each index in the string D, all possible combinations of indices in A, B and C corresponding to match points in matrices $A : B$ and $A : C$ are examined and the number of instances of Equality (1) for that index in D is memorised in matrices $B : D$ and $C : D$. By adding up these values for each character that we know will appear in D, and for each index in D, a character–position matrix can be built, from which the answer can ultimately be decoded.

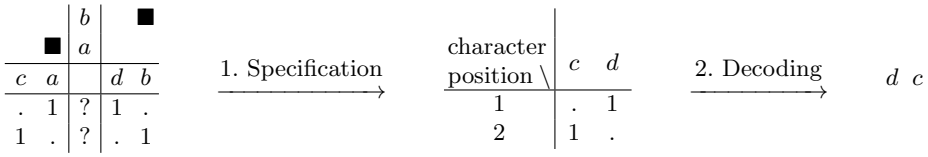

Fig. 1. Specification of the answer of the analogy question $ab : ac :: db : x$ using character–position arithmetic. The characters in the answer and their number of occurrences are known: $\{\, c : 1,\ d : 1 \}$. The black squares (■) in the *upper* part on the *left* visualise the match points in the matrices $A : B$ and $A : C$. The *lower* part gives the number of instances of Equality (1) in each cell of matrices $B : D$ and $C : D$ using the match points and character–position arithmetic. In the *middle*, the character–position matrix summarises the evidence for each character and position (. stands for zero). This character–position matrix can be decoded into the answer dc on the *right*.

2.4 Virtual beginning and end match points

We now turn to a first problematic case where Proposition (6) is not verified, although it should be: work : sing :: you work : you sing. In this case, no triple of indices in A, B and C can be found for the first position in D corresponding to the character y. The character y in D can only come from the same character in C in the first position, as it does not appear in B. However, no character in A is equal to any character in B, so that there is no match point. So Proposition (6) does not hold.

Now, Proposition (6) holds for $\vdash$work$\dashv$: $\vdash$sing$\dashv$ :: $\vdash$you work$\dashv$: $\vdash$you sing$\dashv$ where beginning and end markers are added in the previous example. In that case, for position $i_D = 1$ in D corresponding to character y, the triple of indices $(i_A = 0, i_B = 0, i_C = 1)$ in A, B and C is such that: $D[1 = 0 - 0 + 1] = C[1] =$ y and $A[0] = B[0] = \vdash$. The addition of such markers is tantamount to the insertion of virtual match points in the matrices representing $A : B$ and $A : C$. On our set of positive examples (see Section 5), Proposition (6) holds for all examples when adding such virtual match points

2.5 Match points inside diagonal bands

We turn to a second problematic case for Proposition (6). Taking all possible match points into account may give too much weight for some of them, leading to wrong answers. This is the case for leaf : leaves :: wolf : D. Taking into account all possible match points in the matrices $A : B$ and $A : C$ leads to 10 instances of Equation (1) voting in favour of $D[1] =$ l, and only 7 in favour of $D[3] =$ l (while the situation is balanced for $D[1] =$ w and $D[3] =$ w with 8 equations each). This leads to the incorrect answer 'lowves' instead of 'wolves'.

The work in [4] considers only match points lying on the edit distance traces in $A : B$ and $A : C$, to create the answers to analogy questions. We follow this idea, but not as strictly. In [9, p. 106, illustrated on Figure 3], it is proven that the match points on edit distance traces between strings X and Y lie inside a diagonal band in the edit distance matrix. This diagonal band can be equivalently defined by Inequality (7) which uses the notion of similarity between two strings, i.e., the length of their longest common subsequence, noted s below, instead of the notion of edit distance.

$$-|X| + s(X,Y) \;\leq\; i_Y - i_X \;\leq\; |Y| - s(X,Y) \tag{7}$$

On the previous example, restricting to match points inside diagonal bands delimited by Inequality (7) in all matrices yields more instances of Equality (1) favouring $D[3] =$ l (4 equalities) over $D[1] =$ l (1 equality only), and no more equalities to support $D[3] =$ w; this leads to the correct answer 'wolves'.

2.6 Re-estimating values in the character–position matrix

We turn to a third and last problematic case for Proposition (6). For the analogy question in German *setzen* : *setzte* :: *lachen* : D, the character–position matrix

built as described above somehow hesitates for the but last position between t and e: 2 instances of Equality (1) support $D[5] = t$, 6 support $D[5] = e$, 1 supports $D[6] = t$, and 4 support $D[6] = e$. This leads to the incorrect answer '*lachet*' instead of '*lachte*'.

The situation is similar to the one encountered in statistical machine translation where word-to-word correspondence probabilities should be re-estimated from the mere evidence that they appear in corresponding sentences. The answer consists in using the expectation–maximisation (EM) algorithm to estimate a probability distribution that will maximise the entropy over all possible word-to-word correspondences. The problem here is similar. Words in the source and target languages in machine translation correspond to character and positions in the character–position matrix. When applying the EM algorithm to the character–position matrix on the previous example, the probabilities for the previous characters and positions are re-estimated as follows: $p(D[5] = t) = 0.29$ now exceeds $p(D[5] = e) = 0.25$, and $p(D[6] = e) = 0.32$ exceeds $p(D[6] = t) = 0.28$. This leads to the correct solution '*lachte*'.

3 Decoding the answer of an analogy question

3.1 Solving an assignment problem

In the previous section, we showed how a character–position matrix can be built which assigns a probability to each character and position in the answer D of an analogy question $A : B :: C : D$. The final problem is thus an optimal assignment problem where each position should receive a character and each character should go to a position in D without conflict. This problem can be solved by the Hungarian method, or Kuhn's algorithm [1]. In our setting, we look for a solution of the assignment problem with a maximal cost.

It has also been shown that the Hungarian method is in fact the limit of an entropy maximisation problem [8]. So, we implement a naive and imperfect algorithm which works as follows. We assign each (possibly repeated) character to a position by scanning all characters in increasing order of entropy over all available positions. We assign a character to the position where it gets its highest probability. As several characters may have the same entropy simultaneously, we choose positions so as to avoid conflicts. If conflicts cannot be avoided, we simply stop the process and output no solution for the analogy question. Else, the characters and positions just assigned are removed, the entropies are computed again for the remaining characters and positions, and the process is repeated until all characters have been assigned to a position. If some remaining character cannot be assigned to any position, no solution is output.

This strategy is more prone to fail than the Hungarian method, and should thus be considered as a loose baseline.

3.2 Plurality of answers

There may be no answer, one answer or several answers to an analogy question. For instance, the analogy question $abcabc : gh :: mnkl : D$ has no solution; the

analogy question $\mathit{easy} : \mathit{uneasy} :: \mathit{known} : D$ has only one possible answer: $D = \mathit{unknown}$; and the analogy question $aa : ab :: aaa : x$ has two possible answers only, *aab* or *aba*, if considered as an analogy of commutation (see Footnote 1).

From the solution delivered by the Hungarian method, it is possible to look in the character-position matrix and enumerate all other solutions of same cost by performing all possible exchanges between characters and positions.

4 Overview of the proposed method

4.1 Sketch of the method

Algorithm 1 Solving an analogy question $A : B :: C : x$.

```
def Solve(A, B, C):

    # 1. Specify the answer by a character-position matrix, M.
    ComputeKnownFeatures(A, B, C)
    M[i_D][c_D] = 0 for all c_D ∈ D and i_D ∈ {1,...,|D|}
    for each (i_A,i_B) / InsideDiagonal(i_A,i_B) and A[i_A] == B[i_B]:
        for each (i_C, i_D = i_B − i_A + i_C) / InsideAllDiagonals(i_A, i_B, i_C, i_D):
            M[i_D][C[i_C]] += 1
    #    Do same thing as above by exchanging B and C.
    ...
    M = ExpectationMaximisation(M)
    # 2. Decode the character-position matrix.
    list of pairs (i_D,c_D) = HungarianMethod(M)
    return EnumerateAllSolutions(M, list of pairs (i_D,c_D)))

def ComputeKnownFeatures(A, B, C):
    s(A,B), s(A,C) = similarity(A,B), similarity(A,C)
    s(B,D), s(C,D) = s(A,C) − |A| + |B|, s(A,B) − |A| + |C|
    |D| = |B| − |A| + |C|
    for each character c:
        occ♯_in_D[c] = occ♯_in_B[c] - occ♯_in_A[c] + occ♯_in_C[c]

def InsideDiagonal(i_X, i_Y):
    return −|X| + s(X,Y) ≤ i_Y − i_X ≤ |Y| − s(X,Y)

def InsideAllDiagonals(i_A, i_B, i_C, i_D):
    return all(InsideDiagonal(i_X,i_Y) for (X,Y) in [(A,C), (B,D), (C,D)])
```

Algorithm 1 sketches the method as already illustrated in Figure 1. After computing the features of the answer, a character–position matrix is built and its cells are filled using the character-position arithmetic. The values in the character–position matrix are then re-estimated using the expectation–maximisation algorithm. Decoding is performed using the Hungarian method. This out-

puts one answer. An enumeration of all character–position exchanges of same cost yields all possible answers.

4.2 Complexity analysis of the method

We give a very rough analysis of the complexity of the method. Computation of similarities or enumeration of the match points are basically square in the length of the strings, so that the most costly component in the algorithm is solving the assignment problem by the Hungarian method, known to be cubic in the size of the square matrix, i.e., cubic in the length of the solution in its best implementation. The convergence of the EM algorithm is difficult to estimate[2].

It is interesting to observe that the method is linear in the size of D in the best case, i.e., when $A = B$. In that case, the diagonal band is reduced to the main diagonal in the matrix $A : B$. Consequently, the character–position matrix exhibits a degenerated form where each character in C is assigned the same position in D as in C. Such a matrix is a degenerated case for the Hungarian method, which returns a solution in one pass over the matrix.

5 Experiments

To inspect the accuracy of our proposed method, we use an in-house data set of 160 examples, 113 positive examples and 47 negative examples. Most of the positive examples are from various languages: Arabic, Chinese, English, German, etc. They address complex phenomena, like parallel infixing, as in:

$$\text{(German)} \quad \textit{sprechen} : \textit{ihr ausspr\"achet} :: \textit{nehmen} : \textit{ihr ausn\"ahmet}$$

In addition, some formal positive examples address incrementing problems:

$$\textit{abc} : \textit{abcabc} :: \textit{abcabcabc} : \textit{abcabcabcabc}$$
$$\textit{ab} : \textit{aabb} :: \textit{aaaaaabbbbbb} : \textit{aaaaaaabbbbbbb}$$

The purpose of the negative examples is to test the ability of our method not to deliver an incorrect answer. The negative examples are of the type:

$$\textit{ab} : \textit{aabb} :: \textit{aaabbb} : \textit{aaabbbba}$$

where the answer proposed, *aaabbbba*, is incorrect; the only correct answer, when restricting to analogies of commutation, is *aaaabbbb*. In this case, an algorithm that would blindly output all possible combinations of four a's and four b's in any order would have the incorrect answer in its set of solutions; it would thus fail the test.

We test different combinations of components: for the specification of the answer, use of all possible match points vs. only those inside diagonal bands

[2] We set the convergence threshold to the reciprocal of the number of cells in the character–position matrix, i.e., $1/|D|^2$. In general we observe convergence after a very small number of steps.

(Sect. 2.5), and use of the EM algorithm to re-estimate values in the character–position matrix vs. no use (Sect. 2.6); for decoding, use of the Hungarian method vs. our loose baseline (increasing entropies) (Sect. 3.1).

The results in Table 1 show that each of the components contributes to accuracy. Considering only match points inside diagonal bands allows to jump from below 66 % accuracy to almost 80 % and above. The EM algorithm may be of no utility or may add around 5 % in accuracy. As expected, the Hungarian method always beats our loose baseline by at least 5 % in accuracy. The best accuracy obtained is 91 % when decoding using the Hungarian method.

Match points inside diagonal band	EM algorithm	Decoding method	Recall	Precision	Accuracy
No	No	Increasing entropies	98	41	57
		Hungarian method	98	50	64
	Yes	Increasing entropies	98	41	57
		Hungarian method	98	53	66
Yes	No	Increasing entropies	99	70	78
		Hungarian method	99	82	87
	Yes	Increasing entropies	99	81	86
		Hungarian method	99	88	91

Table 1. Testing 8 different configurations to output the answer specified by character-position arithmetic. The best configuration is the last one.

Table 2 details the confusion matrix for the best configuration. Recall, precision and accuracy are computed in the standard way, and were reported in Table 1. This confusion matrix shows that the weakness of the method lies in too many negative predictions on positive examples. This is measured by a relatively low precision of 88 %.

	Positive predictions	Negative predictions	Total
Positive examples	TP = 99 (62 %)	FN = 14 (9 %)	113 (71 %)
Negative examples	FP = 1 (0 %)	TN = 46 (29 %)	47 (29 %)
Total	100 (63 %)	60 (37 %)	160 (100 %)

Table 2. Confusion matrix for the verification of analogies on 113 positive examples of analogies and 47 negative examples in the best configuration (see Table 1, last line).

This relatively low precision will now be nuanced by results on a much larger dataset which supposedly reflects more real analogy questions. This dataset is from Task 1 of Track 1 of SIGMORPHON 2016 Shared Task: Morphological Reinflection. We use all the offered languages: Arabic, Finnish, Georgian, Ger-

man, Hungarian, Maltese, Navajo, Russian, Spanish and Turkish.[3] We extract analogy questions from such data by considering all analogies of form filtered by morphological features. For each analogy, four different analogy questions can be asked, each of the four terms becoming the answer.[4] As an example in Spanish, the four following questions correspond to the same analogy:

$$\begin{aligned}
&alterado : alterada :: adeudados : x &\Rightarrow\ & x = adeudadas\\
&alterada : alterado :: adeudadas : x &\Rightarrow\ & x = adeudados\\
&adeudadas : adeudados :: alterada : x &\Rightarrow\ & x = alterado\\
&adeudados : adeudadas :: alterado : x &\Rightarrow\ & x = alterada
\end{aligned}$$

The number of analogy questions obtained in each language is given in Table 3. The total number over all languages exceeds 40 million analogy questions. For half of the languages, the percentage of correct answers is equal to or higher than 95 %. The total number of correct answers over all questions in each language reaches 94 %.

Language	Number of analogy questions	% of correct answers
Arabic	381,132	94
Finnish	3,076	95
Georgian	7,256,156	87
German	349,796	91
Hungarian	15,157,368	94
Maltese	10,000	97
Navajo	18,588,020	97
Russian	66,672	99
Spanish	95,564	95
Turkish	729,092	86
Total	42,636,876	94

Table 3. Solving analogy questions extracted from all training data of Task 1 of Track 1 from SIGMORPHON 2016 Shared Task for the best configuration of our proposed method (last line in Table 1)

6 Conclusion and future work

We showed how to answer analogy questions $A : B :: C : D$ of unknown D between strings of characters, by essentially relying on an intuitive basic arithmetic equality: $D[i_B - i_A + i_C] = B[i_B] - A[i_A] + C[i_C]$. We decomposed the problem into two steps: specification and decoding. We performed experiments on a set

[3] `https://github.com/ryancotterell/sigmorphon2016/tree/master/data/`. We use all the files of the type `<language>-task1-train`.

[4] This is not the task proposed in SIGMORPHON Shared Task, which consists in a machine learning task: predicting a word form given a lemma and morphological features after having learnt from the training data.

of positive and negative examples and measured the contribution of each of the components in accuracy. We further assessed the precision of the method on a very large set of more than 40 million analogy questions from the dataset of a shared task in morphology. We obtained the correct answer in 94 % of the cases.

As future direction, we want to carry on in testing the efficiency of the character–position arithmetic. For instance, it remains to inspect whether restricting further to those match points lying on edit distance traces helps or harms and whether we can dispense with the EM algorithm.

References

1. Kuhn, H.W.: The Hungarian method for the assignment problem. Naval Research Logistic Quarterly 2, 83—97 (1955), http://onlinelibrary.wiley.com/doi/10.1002/nav.3800020109/epdf
2. Langlais, P., Patry, A.: Translating unknown words by analogical learning. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL). pp. 877–886 (2007), https://www.aclweb.org/anthology/D/D07/D07-1092.pdf
3. Langlais, P., Yvon, F.: Scaling up analogical learning. In: Proceedings of the 22nd International Conference on Computation Linguistics (Coling 2008). vol. Posters, pp. 51–54. Coling 2008 Organizing Committee, Manchester, UK (August 2008)
4. Lepage, Y.: Solving analogies on words: an algorithm. In: Proceedings of the Association for Computational Linguistics (ACL'98) and 17th International Conference on Computational Linguistics (COLING'98). vol. I, pp. 728–735. Montreal (Aug 1998), https://aclweb.org/anthology/P/P98/P98-1120.pdf
5. Lepage, Y.: De l'analogie rendant compte de la commutation en linguistique. Mémoire d'habilitation à diriger les recherches, Université de Grenoble (May 2003), https://tel.archives-ouvertes.fr/tel-00004372/document
6. Lepage, Y.: Proportional analogy in written language data. In: Gala, N., Rapp, R., Bel-Enguix, G. (eds.) Language, Production, Cognition and the Lexicon, pp. 1–23. Text, Speech and Language Technology 48, Springer International Publishing Switzerland (2014), http://link.springer.com/chapter/10.1007%2F978-3-319-08043-7_10
7. Mikolov, T., Yih, W.T., Zweig, G.: Linguistic regularities in continuous space word representations. In: Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2013). pp. 746–751. Association for Computational Linguistics, Atlanta, Georgia (June 2013), http://www.aclweb.org/anthology/N13-1090
8. Sharify, M., Gaubert, S., Grigori, L.: Solution of the optimal assignment problem by diagonal scaling algorithms. ArXiv e-prints (Apr 2011), https://arxiv.org/pdf/1104.3830.pdf
9. Ukkonen, E.: Algorithms for approximate string matching. Information and Control 64, 100–118 (1985), http://www.sciencedirect.com/science/article/pii/S0019995885800462
10. Yvon, F., Stroppa, N., Miclet, L., Delhay, A.: Solving analogical equations on words. Technical report ENST2004D005, ENST (Jul 2004)