Solving Morphological Analogies Through Generation

Kevin Chan[1,†], Shane P. Kaszefski-Yaschuk[1,†], Camille Saran[1,†], Esteban Marquer[1] and Miguel Couceiro[1,*]

[1]Université de Lorraine, CNRS, LORIA, F-54000, France

Abstract

This contribution is a first attempt at solving morphological analogies through generation, instead of relying on retrieval approaches. Our preliminary experiments show promising results for some languages and reveal the feasibility of the approach in generating solutions of analogical equations in the morphology setting.

Keywords

Morphological analogy, Analogy solving, Representation learning, Word generation

1. Introduction

Analogical proportions are understood as statements of the form "A is to B as C is to D" denoted $A : B :: C : D$, and they are the basis of analogical inference. Analogical inference is a remarkable capability of human reasoning, and that has been used to solve hard reasoning tasks. To some extent, it can be thought of as transferring knowledge from a source domain to a different, but somewhat similar, target domain by relying simultaneously on similarities and dissimilarities. Analogy based reasoning (AR) is closely related to case-based reasoning and has gained increasing interest from the artificial intelligence (AI) community, and has shown its potential in multiple machine learning (ML) tasks such as classification, decision making and recommendation with competitive results [1, 2, 3, 4]. Furthermore, analogical inference can support data augmentation through analogical extension and extrapolation for model learning, especially in environments with few labeled examples [5]. Also, it has been successfully applied to several classical NLP tasks such as machine translation [6], several semantic and morphological tasks [7, 8, 9], as well as (visual) question answering and solving puzzles and scholastic aptitude tests [10, 11].

There are two basic tasks associated with AR. The first is *analogy detection* that corresponds to the task of deciding whether a quadruple A, B, C, D constitutes a valid analogical proportion. This task asks for a common theoretical framework. However, the notion of analogy is not

IARML@IJCAI-ECAI'2022: Workshop on the Interactions between Analogical Reasoning and Machine Learning, at IJCAI-ECAI'2022, July, 2022, Vienna, Austria

*Corresponding author.

†Equal contribution.

kevin.chan3@etu.univ-lorraine.fr (K. Chan); shane-peter.kaszefski-yaschuk5@etu.univ-lorraine.fr (S. P. Kaszefski-Yaschuk); camille.saran5@etu.univ-lorraine.fr (C. Saran); esteban.marquer@loria.fr (E. Marquer); miguel.couceiro@loria.fr (M. Couceiro)

0000-0003-2315-7732 (E. Marquer); 0000-0003-2316-7623 (M. Couceiro)

© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

consensual, and there have been several efforts that follow different axiomatic and logical approaches [12, 13]. For instance, [14] introduces the following 4 postulates in the linguistic context as a guideline for formal models of analogical proportions: *symmetry* (if $A : B :: C : D$, then $C : D :: A : B$), *central permutation* (if $A : B :: C : D$, then $A : C :: B : D$), *strong inner reflexivity* (if $A : A :: C : D$, then $D = C$), and *strong reflexivity* (if $A : B :: A : D$, then $D = B$). Such postulates appear reasonable in the word domain, but they can be criticized in other application domains [15, 16].

The second basic task is *analogy solving* that refers to the task of extrapolating or generating, for a given triple A, B, C the value X such that $A : B :: C : X$ is a valid analogy. One approach to tackling this task is by retrieval and adaptation, *i.e.*, defining an X from a pool of retrieved candidate solutions to be suitably adapted. In fact, analogy solving is somewhat related to case-based reasoning (CBR) [17] where, given a set P of problems, a set S of solutions and a set $\mathcal{C}$ of cases $(x, y) \in P \times S$, the CBR task is to find a solution y_t to a given target problem x_t. CBR basically consists in (1) selecting k source cases in the case base according to some criteria related to the target problem (retrieval step), and (2) reusing the k retrieved cases for proposing a target solution (adaptation step). Despite being a reasonable approach in controlled settings, it suffers from several drawbacks: it requires a suitable choice of examples and is intrinsically limited by case based approaches, that prevent creative inference and innovation.

More recent approaches to analogy solving take advantage of recent deep neural network frameworks that rely on vector representations and on the structure of the underlying multidimensional space. Essentially, analogical proportions are formalized in terms of the *parallelogram rule* by which four vectors e_A, e_B, e_C, and e_D (representing four elements A, B, C, and D) are in analogical proportion if $e_D - e_C = e_B - e_A$. Such an arithmetic view of analogical proportions has been used since the first works on analogy [18], and it was the key element in the methodology employed by earlier neural-based approaches [19, 20]. In the absence of a decoder, the authors implicitly generate a representation e_X and then retrieve the closest candidate D from the vocabulary to solve the analogical equation $A : B :: C : X$ (see brief discussion of Subsection 2.2). However, Chen *et al.* [21] argue that the latter two methods significantly differ from human performance.

In the case of sentence analogies (*i.e.,* where A, B, C are sentences), [22] overcomes this issue by training a decoder that is then used to decode e_X. In this paper, we employ a similar approach in the setting of word analogies. More precisely, following the tracks of [23, 24, 25], we address morphological issues on words and tackle the problem of solving morphological analogies. Inspired by the work of [22] to solving sentence analogies, the novelty in our contribution is to make use of autoencoders to solving morphological analogies on words. More precisely, the main contributions of this paper are as follows: (*i*) we propose a model to generate words at character level from word embeddings with high reconstruction performance, and (*ii*) we achieve encouraging results to solving morphological analogies by generation, thus indicating the feasibility of the approach. Nonetheless, this constitutes ongoing research that requires further investigations.

The paper is organized as follows. We first briefly survey previous work on both main tasks dealing with morphological analogies in Section 2. We then describe the key components of the deep learning architecture as well as the analogy solving procedure we use in Section 3. The empirical setting setting is then presented in Section 4 where we also discuss the experimental

results. We conclude with a general overview of this contribution in Section 5 and propose further directions of future research.

2. Related Approaches

In this paper, we focus on morphological analogies, *i.e.*, analogies on words A, B, C, and D that capture morphological transformations of words (*e.g.*, conjugation or declension). In this section we introduce key approaches of analogy detection and solving in morphology. The main trend follows the seminal work of [26] by exploiting the postulates of analogical proportions mentioned in introduction, but some approaches including ours take a slightly different approach. As deep learning approaches to morphological analogies are strongly related to approaches on semantic word analogies, the latter will also be discussed here.

2.1. Analogy Detection

As mentioned above, the analogy detection task corresponds to classifying quadruples A, B, C, D into valid or invalid analogies. The tools in [27] detect morphological analogies using the number of characters occurrences and the length of the longest common subword. Their approach is designed to generate analogical grids, *i.e.*, matrices of transformations of various words, similar to paradigm tables in linguistics [7]. A data-driven alternative was implemented by [8] for semantic word analogies. Using a dataset of semantic analogies, they learn a neural network to classify quadruples A, B, C, D into valid or invalid analogies, using their embedding e_A, e_B, e_C, and e_D. This approach was applied to morphological analogies in [24] by replacing the GloVe [28] semantic embeddings used by Lim et al. with a morphology-oriented word embedding model.

2.2. Analogy Solving

Approaches to analogy solving usually *generate* the fourth element to solve the analogy, but it is also possible to leverage a list of candidates and *retrieve* the most fitting fourth term to solve the analogy. In Subsubsection 2.2.2 we describe key approaches using the former method to solve morphological analogies, and similarly in Subsubsection 2.2.1 for the latter method. Many approaches in embedding spaces use the latter method because generation from an embedding space can be challenging, and we describe some in Subsubsection 2.2.1. However, such retrieval approaches are limited to the available vocabulary and are unable to perform *analogical innovation*, despite it being a key mechanism in the evolution of languages [29, 30].

2.2.1. Retrieval

Analogy solving on word embeddings has been around since early works on *Latent Semantic Analysis* [31] and word embeddings [20, 23], in which examples like $king - man + woman = queen$ have been used do demonstrate the ability to encode semantic features in the word representation. These examples can be formulated as analogical equations $man : woman :: king : X$, for which the solution is retrieved among a *vocabulary*

of candidate words. In [23], the authors use morphological[1] analogies to demonstrate that some word embedding models encode a degree of morphological information. Two of the most used methods for solving analogies in embedding spaces by retrieval are 3CosAdd [20] and 3CosMul [32]. In 3CosAdd, the solution X is retrieved from the vocabulary by minimizing the cosine distance $cos(e_{word}, e_X)$, with $e_X = e_C - e_A + e_B$ and e_A, e_B, e_C, and e_X the embeddings of A, B, C, and X. 3CosMul follows a similar intuition but we refer the reader to [32] for a detailed description. However, the quality of the solution produced by the methods described above have been criticized by [19] for being far from human performance in some cases. Nonetheless, frameworks based on analogy datasets like those mentioned in [8] appear to bridge this gap in performance. By replacing the arbitrary formula by a learned estimator, Lim et al. significantly improved performance on solving semantic word analogies. This latter approach was adapted to morphological word analogies in [25] and outperforms the generative methods described in Subsubsection 2.2.2. Those two approaches rely on the postulates of analogical proportions, and achieve high analogy solving performance.

While the approach by Marquer et al. has state of the art performance on solving analogical equations in morphology, it suffers from the limitations of retrieval approaches: the solutions are retrieved from a *de facto* finite vocabulary and analogical innovation is impossible. By using a generative deep learning model, the present work aims to maintain state of the art performance while solving the limitation of retrieval approaches.

2.2.2. Generation

In [33], the author uses the postulates of [26] to address multiple characteristics of words, such as their length, the occurrence of letters and of patterns. Based on these features, Lepage proposes an algorithm to solve analogies between character strings. Following the results of [34] about closed form solutions, the *Alea* algorithm [6] proposes a Monte-Carlo estimation of the solutions of an analogical equation by sampling among multiple sub-transformations. Those sub-transformations are obtained by considering the words as bags of characters and generating permutations of characters that are present in B but not in A on one side, and characters of C on the other. Intuitively, if we consider $bag(A)$ the bag of characters in A, *Alea* considers $bag(D) = (bag(B) - bag(A)) + bag(C)$ and thus D is a permutation of the characters of $bag(D)$. Recently, a more empirical approach was proposed by [9], which does not rely on the axioms of analogical proportions. The generation model proposed by the authors considers some transformation f such that $B = f(A)$ and $f(C)$ is computable. The simplest transformation f is usually the one human use to solve analogies [9], and is found by minimizing the Kolmogorov complexity of f. This complexity is estimated by first expressing f using a language of operations (insertion, deletion, *etc.*), and computing the length of the resulting program. Unlike *Alea*, *Kolmo* is able to handle mechanisms like reduplication (repeating part of a word).

Recently, [22] proposed a generation framework to solving sentence analogies. They use an autoencoder model (named ConRNN) trained to reconstruct sentences, and perform simple

[1] In [23] the authors refer to morphological transformations as *syntactic* transformations, because they refer to the syntactic role of the word (*e.g.*, past participle) and not the arrangement of its morphemes (*e.g.*, the addition of the suffix "-ed").

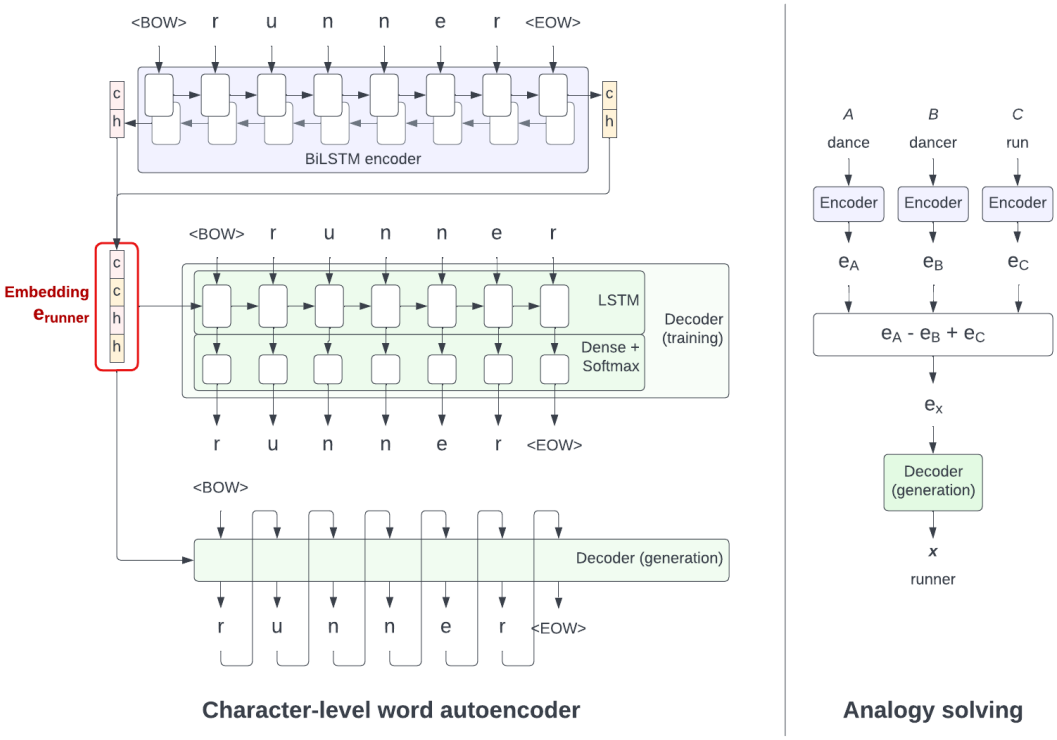

Figure 1: Character-based word auto encoder and vector arithmetic to solve analogies

arithmetic operations on the embedding space to solve analogies. Once the analogy between embeddings is solved, the decoder part of ConRNN is used to generate the solution from the predicted embedding. Their model is a sequence-to-sequence model composed of 2 elements. First, a sentence (as a sequence of words) is used as input to an encoder RNN, and the last hidden state of the RNN is used as the sentence embedding. The latter is then fed to a decoder RNN that tries to predict the words of the input sentence. The use of a generative model achieves significantly better results than previous retrieval approaches on the same embedding space. The current work aims to extend the one of [25] by replacing the retrieval by the generation of the solution of morphological analogical equations. To do so, it is necessary to generate words at the character level from fixed-size embeddings, however in the literature there is to our best knowledge no approach proposed to tackle this specific issue. Inspired by the success of [22], we propose a character-level autoencoder for words and display its performance in solving morphological analogies.

3. Our Approach

In this section we present the approach we use, illustrated in Figure 1. The architecture for our model is a character-level sequence-to-sequence autoencoder model, based on the model

described in [35]. In order to properly decode the final vector solution, the model is trained to encode words and then decode the resulting vector back into the same word. Each character in a word w is encoded into a one-hot vector and is then fed into the encoder, which uses a Bidirectional Long Short Term Memory (BiLSTM) layer. This layer outputs four vectors: the last hidden state h_f and cell state c_f in the forward direction, and similarly h_b and c_b for the backward direction. The concatenation of these vectors $e_w = \text{concat}(h_f, h_b, c_f, c_b)$ is the embedding of the word. The decoder is a regular LSTM layer, followed by a dense layer with softmax activation. The input for the first step of the decoder is the above-mentioned embedding, split into two states $h = \text{concat}(h_f, h_b)$ and $c = \text{concat}(c_f, c_b)$. During training, we use *teacher forcing*: (*i*) the characters of the word w to predict are used as input, with an added *beginning-of-word* (BOW) character at the begining; (*ii*) the prediction targets are the characters of w, but ahead by one time-step and with an *end-of-word* (EOW) character at the end.

To compute the solution of an analogy $A : B :: C : X$, the embeddings e_A, e_B, and e_C are computed by the encoder and used to compute $e_X = e_B - e_A + e_C$. Then, e_X is decoded into a word X by the decoder. Beginning with the BOW character, at each time-step the sampled character with the highest probability of occurrence is added to the word until either the EOW character is predicted or the length of the word is the same as the longest word in the dataset.

4. Experiments

In this section we present our experimental setup. First, the dataset we use is described in Subsection 4.1. We then report the performance of our model in the autoencoder setting in Subsection 4.2. The analogy solving performance of our approach is compared with baselines in Subsection 4.3. Finally, we discuss the overall performance of the model in Subsection 4.4.

4.1. Datasets

For our experiments, we used the analogies from 8 languages available in the Siganalogies dataset [36]: Arabic, English, French, German, Hungarian, Portuguese, Russian, and Spanish extracted from the high resource languages of Sigmorphon2019 [37]. These languages were chosen such that, in later stages of the work, the authors have enough linguistic knowledge to interpret the model outputs. In order to obtain train and test sets, non-overlaping random subsets of analogies from the entire Sigmorphon dataset for a given language are taken, to ensure that no analogy is seen in both the train and test sets.

The Siganalogies dataset also provides a method for data augmentation via permutating the four words in a given analogy. These permutations are obtained using the *symmetry* and *central permutation* postulates of analogy. From a *base form* $A : B :: C : D$, we generate 7 permutations:

- $A : C :: B : D$;
- $D : B :: C : A$;
- $C : A :: D : B$;
- $C : D :: A : B$;

Table 1
Autoencoder accuracy at the word level for 8 languages, trained for 100 epochs on 40,000 random words.

Language	Accuracy (%)
Arabic	99.99
English	99.98
French	99.99
German	99.98
Hungarian	99.97
Portuguese	99.99
Russian	99.96
Spanish	99.98

- $B : A :: D : C$;
- $D : C :: B : A$;
- $B : D :: A : C$.

In Siganalogies, the base forms $A : B :: C : D$ are such that B is an inflected form of A and D is inflected from C. In addition to that, base forms $A : A :: B : B$ derived from the identity postulate ($A : A :: B : B$ is true for all A and B) are present. An example of analogy in English is *dog* : *dogs* :: *cat* : *cats*, another in French is *revérifier* : *revérifiasse* :: *tormenter* : *tormentasse*, and in German there is *Donor* : *Donor* :: *Herstellungsverfahren* : *Herstellungsverfahren* (identity, but also accusative singular declension of the noun).

4.2. Autoencoder performance

As shown in Table 1, our autoencoder achieves very high accuracy in decoding vectors back into words, meaning that any wrong solutions are a result of the operations performed on the analogy rather than the decoding process. In our experiments, the model encodes the words as 128-dimensional vectors and there is a 0.1 dropout on the decoder LSTM layer. The loss function used is categorical cross-entropy, since a probability for the likelihood of each character appearing is required at each time-step. An 80/20 train/validation split is used, and the validation loss was the metric used for the early stopping. If the early stopping is not triggered, the model is trained for 100 epochs.

4.3. Analogy solving performance

Two metrics are used to determine the performance of our model. The first one is a variation of Levenshtein distance, which calculates the minimum number of edits required to change one sequence into another using insertions, deletions, and substitutions. In order to display how close the decoded analogy solutions are to the expected analogy solutions, the Levenshtein distance L was normalized using the length of the manipulated words into a percentage L_p like so:

$$L_p(expected, decoded) = 1 - \frac{L(expected, decoded)}{\max\left(len_{expected}, len_{decoded}\right)}$$

Table 2
Results for 8 languages for 10,000 base analogies and all of their permutations (80,000 analogies in total). We report L_p in % and the accuracy (Acc.) in %. Our autoencoder was trained for 100 epochs on 40,000 random words per language. Baselines Alea [6] and Kolmo [9] were tested in the same setting. The accuracy of the retrieval model ANNr [25] is reported as mean $\pm$ standard deviation for 10 random initialization, but note that these results are not completely comparable with our approach as they were obtained in a closed setting.

Language	Score	Ours	Alea	Kolmo	ANNr
Arabic	L_p	**54.51**	23.72	45.31	–
	Acc.	**12.50**	2.56	3.81	71.80 ± 2.51
English	L_p	**91.58**	88.34	86.75	–
	Acc.	**59.80**	59.65	46.93	94.40 ± 0.67
French	L_p	86.43	80.07	**89.32**	–
	Acc.	51.30	**57.64**	54.49	91.84 ± 0.83
German	L_p	**89.39**	82.76	87.47	–
	Acc.	**52.80**	50.84	48.97	76.95 ± 1.15
Hungarian	L_p	**80.32**	60.72	75.47	–
	Acc.	25.50	**27.80**	23.48	80.42 ± 1.30
Portuguese	L_p	**94.38**	87.97	93.47	–
	Acc.	74.00	**80.06**	71.28	89.30 ± 2.38
Russian	L_p	82.29	63.52	**82.78**	–
	Acc.	33.80	**37.15**	33.44	72.65 ± 1.96
Spanish	L_p	**89.39**	79.49	88.56	–
	Acc.	60.09	**65.02**	58.59	93.01 ± 2.38

The resulting percentage measures the rate of correctly decoded characters per word - when it is 1 (or 100%), then the decoded solution matches the expected solution perfectly. The second metric, accuracy, was calculated by dividing the number of correctly decoded analogies by the total number of analogies for each language. We report the results of decoding a test set of 10,000 base analogies and their permutations in Table 2. We compare our approach with Alea [6] and Kolmo [9] described in Subsubsection 2.2.2. We also report the retrieval accuracy of ANNr [25], however as the model is a retrieval approach it is not directly comparable to our model and other baselines. Instead, it indicates the performance one can reach when bypassing the issue of generation. Our model reaches comparable performance to the generation baselines in terms of L_p for all languages, and comparable performance in terms of accuracy for half of the languages.

4.4. Discussion

The performance on Arabic of all generation models is very low, while the retrieval model does not appear to suffer from the same effect. Further analysis of the data reveals that the character encoding used for Arabic decomposes each character into multiple encoded characters, resulting

in longer and more complex sequences of characters than expected. We suppose this makes generation harder and is the cause of this low performance.

There is a significant difference in the performance of the model depending on which permutations are used. Due to the high accuracy of the decoder and the nature of the parallelogram rule, the model performs very well on analogies where the solution D is the same as another element in the analogical equation. Permutations of this form include strong reflexivity, strong inner reflexivity, and identity.

As Table 2 shows, the raw accuracy is often lower than the baselines Alea and Kolmo, but the Levenshtein percentage is often on par or higher. This suggests that more individual characters are correctly decoded with our model on average when compared to the baselines, but that it does not decode entire words with 100% accuracy as often. This is to be expected as the model is not trained to solve analogies, but rather is trained to properly decode words after vector arithmetic is performed on the encoded vectors.

When applied to the encoded vectors, the parallelogram rule is highly accurate with regular morphology and with certain permutations, but it often struggles when the morphology is more irregular. Since the model decodes individual words with high accuracy, the problem lies with the operations performed on the three vectors in an analogical equation after encoding. Given the model's current performance without explicitly encoding any morphological features or features of analogical equations when training, we expect that better performance can be obtained if these features are included in future iterations of the trained autoencoder.

5. Conclusion and Perspectives

In this paper we proposed an autoencoder framework to solving morphological analogies by generating solutions. This partially addresses the limitations of previous works relying on case based approaches that prevent creative inference and innovation. Our adaptation to the morphology setting was illustrated in several languages with promising results, and that reveal new potential directions for future work.

However, this is a preliminary proposal that will profit from further training and the combination with state of the art retrieval approaches such as ANNr from [25]. Moreover, we will also explore its tranferability potential and its generalization across multiple modalities and data contexts.

Acknowledgments

This research work was partially supported by TAILOR, a project funded by EU Horizon 2020 research and innovation program under GA No 952215, and the Inria Project Lab "Hybrid Approaches for Interpretable AI" (HyAIAI).

References

[1] M. A. Fahandar, E. Hüllermeier, Learning to rank based on analogical reasoning, in: AAAI, 2018, pp. 2951–2958.

[2] M. A. Fahandar, E. Hüllermeier, Analogical embedding for analogy-based learning to rank, in: IDA, volume 12695 of *LNCS*, Springer, 2021, pp. 76–88.
[3] N. Hug, H. Prade, G. Richard, M. Serrurier, Analogical proportion-based methods for recommendation - first investigations, Fuzzy Sets Systems 366 (2019) 110–132.
[4] M. Mitchell, Abstraction and analogy-making in artificial intelligence, Ann. N.Y. Acad. Sci. 1505 (2021) 79–101.
[5] M. Couceiro, N. Hug, H. Prade, G. Richard, Analogy-preserving functions: A way to extend boolean samples, in: 26th IJCAI, 2017, pp. 1575–1581.
[6] P. Langlais, F. Yvon, P. Zweigenbaum, Improvements in analogical learning: Application to translating multi-terms of the medical domain, in: 12th EACL, ACL, 2009, pp. 487–495.
[7] R. Fam, Y. Lepage, Morphological predictability of unseen words using computational analogy., in: 24th ICCBR workshops, 2016, pp. 51–60.
[8] S. Lim, H. Prade, G. Richard, Solving word analogies: A machine learning perspective, in: 15th ECSQARU, volume 11726, 2019, pp. 238–250.
[9] P.-A. Murena, M. Al-Ghossein, J.-L. Dessalles, A. Cornuéjols, Solving analogies on words based on minimal complexity transformation, in: 29th IJCAI, 2020, pp. 1848–1854.
[10] F. Sadeghi, C. L. Zitnick, A. Farhadi, Visalogy: Answering visual analogy questions, in: NeurIPS, 2015, pp. 1882–1890.
[11] J. Peyre, I. Laptev, C. Schmid, J. Sivic, Detecting unseen visual relations using analogies, in: IEEE ICCV, 2019, pp. 1981–1990.
[12] Y. Lepage, Analogy and formal languages, in: 6th CFG and 7th CML, volume 53, 2001, pp. 180–191.
[13] L. Miclet, S. Bayoudh, A. Delhay, Analogical dissimilarity: Definition, algorithms and two experiments in machine learning, JAIR 32 (2008) 793–824.
[14] Y. Lepage, De l'analogie rendant compte de la commutation en linguistique, Habilitation à diriger des recherches, Université Joseph-Fourier - Grenoble I, 2003.
[15] C. Antic, Analogical proportions (2022).
[16] N. Barbot, L. Miclet, H. Prade, Analogy between concepts, Artificial Intelligence 275 (2019) 487–539.
[17] J. Lieber, E. Nauer, H. Prade, When Revision-Based Case Adaptation Meets Analogical Extrapolation, in: 29th ICCBR, volume 12877 of *LNCS*, 2021, pp. 156–170.
[18] D. E. Rumelhart, A. A. Abrahamson, A model for analogical reasoning, Cognitive Psychology 5 (1973) 1–28.
[19] A. Drozd, A. Gladkova, S. Matsuoka, Word embeddings, analogies, and machine learning: Beyond king - man + woman = queen, in: 26th COLING, 2016, pp. 3519–3530.
[20] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, in: 1st ICLR, Workshop Track, 2013.
[21] D. Chen, J. C. Peterson, T. Griffiths, Evaluating vector-space models of analogy, in: 39th CogSci, Cognitive Science Society, 2017, pp. 1746–1751.
[22] L. Wang, Y. Lepage, Vector-to-sequence models for sentence analogies, in: ICACSIS, 2020, pp. 441–446.
[23] T. Mikolov, W.-T. Yih, G. Zweig, Linguistic regularities in continuous space word representations, in: NAACL, 2013, pp. 746–751.
[24] S. Alsaidi, A. Decker, P. Lay, E. Marquer, P.-A. Murena, M. Couceiro, A neural approach

for detecting morphological analogies, in: IEEE 8th DSAA, 2021, pp. 1–10.
[25] E. Marquer, S. Alsaidi, A. Decker, P.-A. Murena, M. Couceiro, A Deep Learning Approach to Solving Morphological Analogies, 2022. URL: https://hal.inria.fr/hal-03660625.
[26] Y. Lepage, S. Ando, Saussurian analogy: a theoretical account and its application, in: 16th COLING, 1996.
[27] R. Fam, Y. Lepage, Tools for the production of analogical grids and a resource of n-gram analogical grids in 11 languages, in: 11th LREC, ELRA, 2018, pp. 1060–1066.
[28] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation, in: EMNLP, 2014, pp. 1532–1543.
[29] D. L. Fertig, Analogy and morphological change, Edinburgh University Press, 2013.
[30] E. Mattiello, Analogy in Word-formation, De Gruyter Mouton, 2017.
[31] S. T. Dumais, G. W. Furnas, T. K. Landauer, S. Deerwester, R. Harshman, Using latent semantic analysis to improve access to textual information, in: SIGCHI, 1988, pp. 281–285.
[32] O. Levy, Y. Goldberg, Dependency-based word embeddings, in: 52nd ACL (Volume 2: Short Papers), ACL, 2014, pp. 302–308.
[33] Y. Lepage, Character-position arithmetic for analogy questions between word forms, in: 25th ICCBR workshops, volume 2028, 2017, pp. 23–32.
[34] F. Yvon, Finite-state transducers solving analogies on words, Rapport GET/ENST&LTCI (2003).
[35] F. Chollet, Character-level recurrent sequence-to-sequence model, 2017. URL: https://keras.io/examples/nlp/lstm_seq2seq/.
[36] E. Marquer, M. Couceiro, S. Alsaidi, A. Decker, Siganalogies - morphological analogies from Sigmorphon 2016 and 2019, 2022.
[37] A. D. McCarthy, E. Vylomova, S. Wu, C. Malaviya, L. Wolf-Sonkin, G. Nicolai, C. Kirov, M. Silfverberg, S. J. Mielke, J. Heinz, R. Cotterell, M. Hulden, The SIGMORPHON 2019 shared task: Morphological analysis in context and cross-lingual transfer for inflection, in: 16th CRPPM workshops, ACL, 2019, pp. 229–244.