A Deep Learning Approach to Solving Morphological Analogies*

Esteban Marquer[1][0000−0003−2315−7732], Safa Alsaidi[1][0000−0002−4132−1068], Amandine Decker[1][0000−0001−6773−9983], Pierre-Alexandre Murena[2][0000−0003−4586−9511], and Miguel Couceiro[1][0000−0003−2316−7623]

[1] Université de Lorraine, CNRS, LORIA, F-54000, France
{esteban.marquer,miguel.couceiro}@loria.fr
[2] HIIT, Aalto University, Helsinki, Finland
pierre-alexandre.murena@aalto.fi

Abstract. Analogical proportions are statements of the form "A is to B as C is to D". They support analogical inference and provide a logical framework to address learning, transfer, and explainability concerns. This logical framework finds useful applications in AI and natural language processing (NLP). In this paper, we address the problem of solving morphological analogies using a retrieval approach named ANNr. Our deep learning framework encodes structural properties of analogical proportions and relies on a specifically designed embedding model capturing morphological characteristics of words. We demonstrate that ANNr outperforms the state of the art on 11 languages. We analyze ANNr results for Navajo and Georgian, languages on which the model performs worst and best, to explore potential correlations between the mistakes of ANNr and linguistic properties.

Keywords: Analogy solving · Neural networks · Retrieval · Morphological word embeddings.

1 Introduction

An analogy, or analogical proportion, is a relation between four elements A, B, C, and D meaning "A is to B as C is to D", often written as $A : B :: C : D$. For example, *cat* : *kitten* :: *dog* : *puppy* and *cat* : *cats* :: *dog* : *dogs* are two analogies between words. Notice that the transformation between *cat* and *kitten* is a semantic one (*kitten* is a young *cat*) while the one between *cat* and *cats* is morphological (*cats* is the plural of *cat*, obtained with the suffix "-*s*"). We call the first example a *semantic analogy* and the second a *morphological analogy*. It can be argued that morphological analogies are covered by semantic analogies, but morphological analogies can be manipulated even without knowledge

* This research 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).

about the semantics of a word. For example, $cat : cats :: Balrog : Balrogs$ is understandable even without knowing what a $Balrog$ is.

There are two main tasks associated with analogical proportions. One is *analogy detection* that refers to task of deciding whether a quadruple A, B, C, D forms a valid analogy $A : B :: C : D$. The other is *analogy solving* that refers to the task of finding possible values x for which $A : B :: C : x$ constitutes a valid analogy. Expressions $A : B :: C : x$ for an indeterminate x are referred to as *analogical equations*. We further detail the two tasks and related approaches to tackle them in Section 2.

Analogy solving can be seen as a form of case-based reasoning (CBR) [20]. Indeed, in CBR the goal is to find a solution s_t to a problem p_t using a set of problems P, a set of solutions S, and cases $(p, s) \in P \times S$ associating a problem p with one of its solutions s. The first step, called *retrieval*[3], is to select k cases in the case base according to some criteria related to p_t. The k retrieved cases are then used to propose a target solution s_t, this is the *adaptation* step. When $k = 1$, we can formulate the adaptation step as solving the analogical equation $p_1 : s_1 :: p_t : x$, with s_t one of the equation's solution. There are other ways to relate analogies and CBR, albeit less related to this paper. For example, if $k = 3$ one can look at cases where $p_1 : p_2 :: p_3 : p_t$ is valid and $s_1 : s_2 :: s_3 : x$ is solvable in S [20]. Like above, s_t would be solution to the equation.

As mentioned in the first paragraph, analogies can subsume relations of different nature (semantic or morphological in the example, but one could build syntactic analogies on sentences or geometric analogies on numbers or images). In this article, we focus on solving morphological analogies. While recent approaches to tackle this problem using symbolic methods achieve state of the art performance on analogy solving [11,14,25], they are inherently restricted to a subset of morphology as they are not able to account for "irregular" cases like $cat : cats :: child : children$. We use deep learning to learn morphological features of words to address this issue. Following a similar idea, Lim *et al.* [21] proposed a deep learning model to tackle semantic analogies on words, based on the analogical framework of Miclet *et al.* [23] and Prade and Richard [27]. Unlike previous works in deep learning, the architecture integrates the characteristics of analogies by design and relies heavily on pretrained GloVe embeddings [26] to compute word representations. We argue that their approach, which already deals with analogies on words, can be applied to morphological analogies. As mentioned in our previous work [1], although the embedding model used by Lim *et al.* is well suited for semantic analogies, it results in lower performance on the morphological counterpart. In our previous work [1] we addressed morphological analogy detection by proposing an embedding model targeting word morphology, and we extend those results for analogy solving in the present work.

To summarize, the main contributions of this paper are the following: (*i*) we demonstrate the importance of well suited embedding space for analogy manipulation, and provide empirical evidence that analogy detection is helpful in obtaining a well suited embedding model for analogy solving, and (*ii*) we pro-

[3] In this article we use *retrieval* in the general meaning and not the one of CBR.

pose a deep learning framework that outperforms state of the art approaches to solve analogies that capture morphological features.

In the remainder of this paper we first introduce the problem of morphological analogies with a brief state of the art in Section 2. We describe the three components of our retrieval approach (ANNc, ANNr and the morphological embedding model) in Section 3, and detail our experiments in Section 4. First, we describe the dataset used in Subsection 4.1, then compare performance with different training procedures and against baselines in Subsections 4.2 to 4.4. Finally, in Subsection 4.5, we explore mistakes made by ANNr on Navajo and Georgian to get insight on the behavior and flaws of our approach. As per tradition, we conclude our paper and offer perspectives on some key aspects in Section 5.

2 The Problem of Morphological Analogy

An *analogical proportion* is a 4-ary relation denoted by $A : B :: C : D$ and interpreted as "A is to B as C is to D". 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).

Analogical proportions. The importance of analogy in morphology has been long known [29], and it has been mathematically formalized following the early works of Lepage [16,18]. These works paved the way towards the current view of *analogical proportions* [23,27] as a 4-ary relation that verifies the following postulates for all A, B, C, and D: reflexivity ($A : B :: A : B$), symmetry ($A : B :: C : D \rightarrow C : D :: A : B$), and central permutation ($A : B :: C : D \rightarrow A : C :: B : D$). It is not difficult to verify that other equivalent forms can be derived using symmetry and central permutation: $D : B :: C : A$, $C : A :: D : B$, $B : A :: D : C$, $D : C :: B : A$ and $B : D :: A : C$. The two main trends in the research on analogies, namely *analogy detection* and *analogy solving*, are described below.

Analogy detection. The analogy detection task corresponds to classifying quadruples $\langle A, B, C, D\rangle$ into valid or invalid analogies. In particular, in the context of semantic analogies, Bayoudh *et al.* [3] proposed to use Kolmogorov complexity as a distance measure between words in order to define a conceptually significant analogical proportion and to classify quadruples as valid or invalid analogies. A data-driven alternative was implemented by Lim *et al.* [21], by learning the classifier directly as a neural network. In particular, analogy detection has been used in the context of analogical grids [11], *i.e.*, matrices of transformations of various words, similar to paradigm tables in linguistics [10].

Analogy solving. The analogy solving task corresponds to inferring the fourth term that makes an analogy valid. This task is by far the most investigated in the analogy literature and several methodologies have been proposed to solve it. The approaches in the next paragraph *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. Many approaches in embedding spaces, including ours, use the latter method as generation from an embedding space can be challenging. In our case the candidates are the set of all the words in the dataset, called the *vocabulary*.

In the context of solving morphological analogies, the main trend follows the seminal work of Lepage and Ando [18] by exploiting the aforementioned postulates of analogical proportions. For instance, Lepage [17] uses these postulates to address multiple characteristics of words, such as their length, the occurrence of letters and of patterns. Following the results of Yvon [32] about closed form solutions, the *Alea* algorithm [14] proposes a Monte-Carlo estimation of the solutions of an analogical equation by sampling among multiple sub-transformations. Recently, a more empirical approach was proposed by Murena *et al.* [25], which does not rely on the axioms of analogical proportions. Their method consists in finding the simplest among all possible transformations from A to B which could also apply to C, using the Kolmogorov complexity of the transformations. We provide more details on the latter two approaches in Subsection 4.3.

Analogy in embedding spaces. A particular and noteworthy form of analogical proportions in vector spaces is the *parallelogram rule* that states that four vectors e_A, e_B, e_C, and e_D are in analogical proportion if $e_D - e_C = e_B - e_A$. This relation has been used since the first works on analogy [28], and it is a key element in the methodology employed by the recent neural-based approaches [8,24]. In this setting, the parallelogram rule is applied in some learned embedding space, with e_A, e_B, e_C, and e_D being the embeddings of four elements A, B, C, and D. We detail two of the most used methods for solving analogies in embedding spaces, namely 3CosAdd [24] and 3CosMul [19], in Subsection 4.3. These two approaches implicitly generate a solution e_x and retrieve the closest candidate from the vocabulary. However, Chen *et al.* [5] argue that the latter two methods significantly differ from human performance. Nonetheless, frameworks based on analogy datasets like those mentioned in [21] appear to bridge this gap in performance. Similarly to the method of Lim *et al.* [21], ours learns a model for both the analogy detection and solving tasks in morphology. This approach models analogy as described by the analogy data, unlike 3CosAdd and 3CosMul that rely on fixed formulas and on the expected properties of the embedding space.

3 Proposed Approach

In this section we present the approach we use, illustrated in Figure 1. There are 3 components in our framework, and they are described in Subsection 3.1: a classification model (ANNc) and a retrieval model (ANNr), both taking as input embeddings computed by a morphological embedding model. We then detail our training and evaluation procedures in Subsection 3.2. Additional information about the parameters of our model are given in Subsection 4.2.

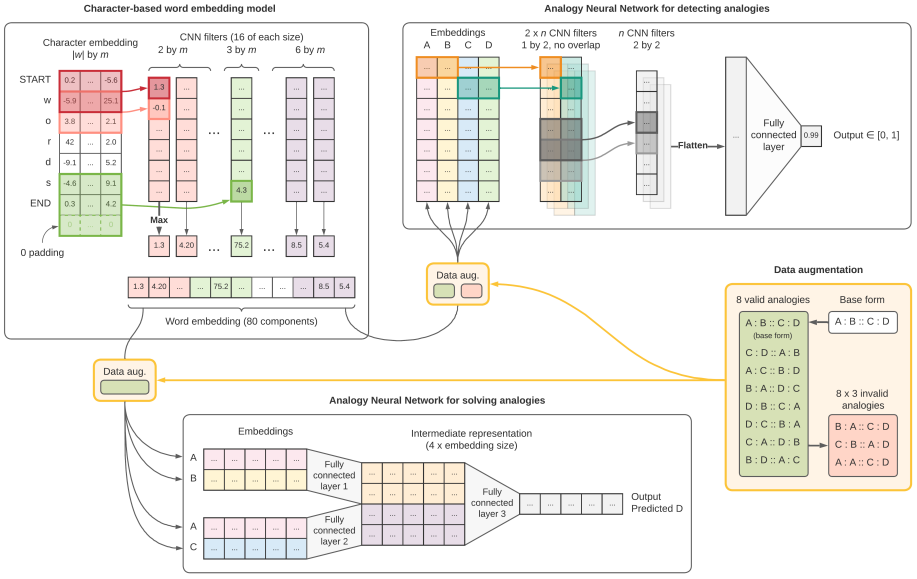

Fig. 1: Morphological embedding model, data augmentation, analogy classification (ANNc) and analogy retrieval (ANNr) models.

3.1 Classification, Retrieval and Embedding Models

The models described in this subsection follow the structure of the ones proposed in [1,21] and are schematized in Figure 1.

Classification. The neural network classifier follows the idea that an analogy $A : B :: C : D$ is valid if A and B differ in the same way as C and D. We call this model *Analogy Neural Network for classification* (ANNc). Technical details on this model can be found in our previous works [1,2].

Retrieval. An approach to solving $A : B :: C : \mathbf{x}$ is to find how e_B differs from e_A and to generate an $e_\mathbf{x}$ that differs from e_C in the same way. Central permutation (see Section 2) allows us to apply the same operations on $A : C :: B : \mathbf{x}$, to obtain $e_\mathbf{x}$ from e_B using the difference between e_A and e_C.

Following this intuition, the model applies two separate fully connected layers with ReLU activation to determine the relation between e_A and e_B on the one side (*fully connected layer 1* in Figure 1) and the one between e_A and e_C on the other side (*fully connected layer 2*). Those layers also encode the necessary information from e_B and e_C, respectively, while taking e_A into account. Finally, *fully connected layer 3* (with no activation function) generates $e_\mathbf{x}$ the embedding of the predicted $\mathbf{x}$ from all the extracted information. We refer to this model as the *Analogy Neural Network for retrieval* (ANNr).

Embedding model. Most deep learning methodologies rely on real-valued vector representations of the data called *embeddings*. For many applications, there are embedding models available and that have been pretrained on large amounts of data, as it saves users the cost of fully training them. It is commonly accepted that the quality of such representations, which corresponds to the amount and nature of the encoded information as well as the properties of the embedding space, is a key factor to achieve higher performance. In our previous work [1] and early experiments with pretrained word embeddings models (*fasttext* [4], GloVe [26], Bert Multilingual [7] and *word2vec* [24]) ANNc performed poorly. Note that *fasttext*, which uses sub-words instead of words and is thus closer to morphology, produced better results. Such general-purpose embedding models, oriented towards the distributional semantics of words, seem ill-suited to manipulate morphological analogies. We thus proposed in [1] a word embedding model inspired from the work of Vania [30], that is more suited to morphological tasks.

As schematized in Figure 1, the inputs of our embedding model are the characters of a word, which are embedded into vectors of size m using a learned embedding matrix. Multiple CNN filters go over the character embeddings, spanning over the full embeddings of 2 to 6 characters, and resulting in filter sizes between 2 by m and 6 by m. For each filter, the model computes the maximum output to serve as a component of the word embedding. This last operation keeps only salient patterns detected by each of the filters, and forces each CNN filter to specialize in identifying a specific pattern of characters. As we use character embeddings to encode character features, the model can capture patterns such as "-ing" but also "vowel-vowel-consonant". This flexibility is useful to deal with phenomena like *euphony* which is, in very simple terms, a change of sound to make a word easier to pronounce (*e.g.*, *far* becomes *further* when adding the suffix *-ther*). These character patterns correspond to *morphemes*, which are the minimal units of morphology. For further details see [1].

3.2 Training and Evaluation

Data augmentation. The postulates of analogical proportions described in Section 2 enable the generation multiple analogies from each quadruple in our analogy dataset. We refer to this process as data augmentation. It extends the amount of data available, but also makes the models learn how to fit the formal postulates of analogy.

Given a valid analogy $A : B :: C : D$, we generate 7 more valid analogies, namely, $A : C :: B : D$, $D : B :: C : A$, $C : A :: D : B$, $C : D :: A : B$, $B : A :: D : C$, $D : C :: B : A$, $B : D :: A : C$. In practice, data augmentation is applied after the embedding model as shown in Figure 1. Further details on the training of the classifier can be found in [1].

Training criterion. As in previous works [1,2,21], we use the Binary Cross-Entropy (BCE) loss to train the classification model. For the retrieval model

(to solve analogies), we experiment with multiple training objectives designed to avoid a collapse of the embedding space when training the embedding model[4].

Our first loss is based on the *Cosine Embedding Loss* (CEL):

$$\mathrm{CEL}(e_D, \widehat{e_D}, y) = \begin{cases} 1 - \cos(e_D, \widehat{e_D}), & \text{if } y = 1 \\ \max(0, \cos(e_D, \widehat{e_D})), & \text{if } y = -1 \end{cases} \tag{1}$$

where e_D is the embedding of D and $\widehat{e_D}$ a prediction for e_D. Here, $y = 1$ if e_D and $\widehat{e_D}$ are to be close, and $y = -1$ otherwise. Given an analogy $A : B :: C : D$, we encourage $\widehat{e_D}$ to be close to the actual e_D and far from e_A, e_B, and e_C the embeddings of A, B, and C by using the sum:

$$\begin{aligned} \mathcal{L}_{\mathrm{CEL}} = \mathrm{CEL}(e_A, \widehat{e_D}, -1) + \mathrm{CEL}(e_B, \widehat{e_D}, -1) \\ + \mathrm{CEL}(e_C, \widehat{e_D}, -1) + \mathrm{CEL}(e_D, \widehat{e_D}, 1) \end{aligned} \tag{2}$$

We follow a similar intuition for two losses based on the L^2 distance or *Mean Squared Error* (MSE):

$$MSE(e_D, \widehat{e_D}) = \frac{1}{n} \sum_{i \in [1,n]} (e_{D,i} - \widehat{e_{D,i}})^2 \tag{3}$$

where $e_{D,i}$ is the i-th component of e_D and n is the number of dimensions of the embeddings. We encourage $\widehat{e_D}$ to be close to the actual e_D and far from other embeddings by normalizing $\mathrm{MSE}(e_D, \widehat{e_D})$ based on distances between other embeddings. Normalizing using the full distribution of distances between embeddings would be very accurate, but computing embeddings of the full vocabulary for each training step would be computationally heavy.

Instead, we experiment with two smaller-scale normalization. A first option is to normalize with regards to the possible pairs of elements in the quadruple:

$$\mathcal{L}_{\text{norm. other}} = \frac{1 + 6 \times MSE(e_D, \widehat{e_D})}{1 + \sum_{a,b \in \{(A,B),(A,C),(A,D),(B,C),(B,D),(C,D)\}} MSE(e_a, e_b)} \tag{4}$$

A second option is to normalize with regards to some random embedding e_z:

$$\mathcal{L}_{\text{norm. random}} = \frac{1 + MSE(e_D, \widehat{e_D})}{1 + MSE(e_z, \widehat{e_D})} \tag{5}$$

As the model is trained with batches of randomly selected samples, we associate each prediction with an unrelated embedding e_z by permuting e_D along the batch dimension.

Finally, we consider the sum of all the above loss terms, in order to determine whether their combination overcomes the limitations of each:

$$\mathcal{L}_{\text{all}} = \mathcal{L}_{\mathrm{CEL}} + \mathcal{L}_{\text{norm. other}} + \mathcal{L}_{\text{norm. random}} \tag{6}$$

[4] An example of embedding space collapse: moving all the embeddings in a smaller area of the embedding space by multiplying them by 10^5 minimizes $\mathrm{MSE}(e_D, \widehat{e_D})$ but does not improve retrieval performance, as the relative distance between embeddings does not change.

Inference and evaluation. The output of ANNr is a real-valued vector in the embedding space, but not necessarily the exact embedding of a word. As the model is not equipped with a decoder for the embedding space, we do not have direct access to the word corresponding to a generated embedding. Instead, we use cosine distance[5] to retrieve the closest embeddings among the embeddings of the whole vocabulary as the model prediction. To evaluate the performance we use top-k retrieval accuracy.

4 Experiments

In this section, we describe some of our experiments on the retrieval model. The code can be found on GitHub[6]. In Subsection 4.1, we briefly describe the dataset used and explain our choice of training procedure in Subsection 4.2. The performance of our model is compared with state of the art baselines in Subsection 4.3, and we explore how far the model prediction is from the expected output in Subsection 4.4. Model mistakes on Navajo and Georgian are studied in Subsection 4.5. Our experiments are implemented in PyTorch and PyTorch Lightning, and were carried out using computational clusters equipped with GPUs from the Grid'5000 testbed (*grele* and *grue* clusters, see `https://www.grid5000.fr`).

4.1 Data

For our experiments, we used the analogies from 11 languages available in the Siganalogies dataset [22]: Spanish, German, Finnish, Russian, Turkish, Georgian, Navajo, Arabic (Romanized), Hungarian, and Maltese extracted from Sigmorphon2016 [6], and Japanese from the Japanese Bigger Analogy Test Set [12]. It is noteworthy that we use the data of the original Sigmorphon2016 dataset, which produces a different set of analogies than the one used by [11,25]. In particular, Siganalogies contains analogies that are hard to solve for a non-data-driven approach (*e.g.*, "be : was :: go : went").

For training and evaluation we use the original split of the Sigmorphon2016 [6] dataset, from which we extract the analogies independently. For the Japanese Bigger Analogy Test Set [12] however, no such split is available and so we split[7] the extracted analogies into 70% for the training, 5% for the development, and 25% for the test set. To maintain reasonable training and evaluation times, we used up to 50000 analogies of each set, randomly selected when loading the data.

4.2 Refining the Training Procedure

We experiment with the four losses mentioned in Subsection 3.2. We also consider two training procedures: we transfer the embedding model learned in the

[5] We experimented with both Euclidean and cosine distance, the former giving slightly better results in most cases, even though the difference is not significant.

[6] `https://github.com/EMarquer/nn-morpho-analogy-iccbr`

[7] The samples were randomly selected using a fixed random seed.

Table 1: Top-1 accuracy (*i.e.*, precision, in %) of ANNr when training for 50 epochs with the embedding from ANNc and from scratch. We report mean ± std. over 10 random seeds. Boldface results are the best for each language.

	$\mathcal{L}_{\text{CEL}}$	$\mathcal{L}_{\text{norm. other}}$	$\mathcal{L}_{\text{norm. random}}$	$\mathcal{L}_{\text{all}}$
Arabic (from scratch)	8.53 ± 0.89	16.01 ± 4.20	27.90 ± 4.66	16.45 ± 4.35
Arabic (transfer)	6.86 ± 1.08	68.84 ± 5.54	$\mathbf{73.08 \pm 4.19}$	68.59 ± 3.17
Finnish (from scratch)	21.31 ± 2.31	64.14 ± 7.07	47.25 ± 9.82	57.35 ± 5.46
Finnish (transfer)	6.23 ± 1.27	87.37 ± 2.11	$\mathbf{90.15 \pm 1.42}$	89.90 ± 1.61
Georgian (from scratch)	34.14 ± 3.44	76.33 ± 7.92	68.56 ± 9.85	77.66 ± 5.42
Georgian (transfer)	16.82 ± 3.33	95.64 ± 1.12	$\mathbf{97.12 \pm 0.26}$	95.09 ± 0.65
German (from scratch)	29.31 ± 3.42	64.05 ± 11.12	51.95 ± 11.21	54.91 ± 8.81
German (transfer)	14.17 ± 2.02	90.66 ± 0.66	91.47 ± 0.55	$\mathbf{91.52 \pm 0.62}$
Hungarian (from scratch)	24.35 ± 1.64	50.11 ± 4.72	41.25 ± 9.39	55.54 ± 4.46
Hungarian (transfer)	11.61 ± 1.93	$\mathbf{89.85 \pm 1.52}$	89.42 ± 1.43	88.79 ± 1.78
Japanese (from scratch)	23.33 ± 2.71	9.76 ± 2.52	34.00 ± 8.54	11.46 ± 3.54
Japanese (transfer)	26.89 ± 1.66	83.10 ± 0.92	$\mathbf{86.12 \pm 0.76}$	74.14 ± 1.09
Maltese (from scratch)	5.70 ± 0.88	48.21 ± 9.24	72.74 ± 3.52	51.82 ± 10.22
Maltese (transfer)	5.63 ± 0.84	95.94 ± 0.73	$\mathbf{97.16 \pm 0.30}$	91.69 ± 1.55
Navajo (from scratch)	6.81 ± 0.69	19.36 ± 3.07	23.66 ± 2.99	24.69 ± 2.49
Navajo (transfer)	6.34 ± 0.64	$\mathbf{53.73 \pm 1.58}$	52.22 ± 1.18	52.45 ± 1.97
Russian (from scratch)	12.21 ± 1.72	38.46 ± 4.77	36.80 ± 3.84	41.04 ± 4.00
Russian (transfer)	4.47 ± 0.68	74.08 ± 1.24	71.66 ± 0.76	$\mathbf{76.12 \pm 1.11}$
Spanish (from scratch)	26.76 ± 1.70	84.26 ± 2.93	78.31 ± 5.91	81.76 ± 4.19
Spanish (transfer)	20.67 ± 4.32	91.03 ± 1.63	$\mathbf{92.63 \pm 1.25}$	91.12 ± 1.65
Turkish (from scratch)	13.30 ± 1.07	31.71 ± 4.40	42.82 ± 5.49	39.16 ± 6.82
Turkish (transfer)	7.41 ± 1.63	86.17 ± 1.66	88.36 ± 1.17	$\mathbf{88.45 \pm 1.12}$

ANNc task, and we compare the results with training the embedding model from scratch with ANNr to confirm the benefits of transferring. When transferring, we obtained the best results by first freezing[8] the embedding model and training until ANNr converges[9], then unfreezing the embedding model and resuming training. We limited the total training time to 50 epochs for ANNr, and 20 for ANNc. For Japanese which has less than $50,000$ training analogies ($18,487$), we limit ANNr training to 150 epochs. In average, half of the training epochs are needed to converge with the embedding frozen, but this varies between runs and ranges between 11 and 42 epochs out of 50 (18 and 68 for Japanese). Comparative results are reported in Table 1.

We use a character embedding size (m in Figure 1) of 64 and 16×5 filters spanning over 2 to 6 input characters, resulting in embeddings of 80 dimensions. We use 128 filters for ANNc (n in Figure 1). We use Adam [13] with the recommended learning rate of 10^{-3}. For retrieval, we use a learning rate of 10^{-5} for the embedding model which produced better results experimentally in both transfer and non-transfer settings.

[8] By freezing we mean that the parameters of the model are not updated.
[9] By convergence we mean that there is no improvement in the development set loss.

Transferring embeddings from ANNc significantly improves ANNr performance. A first conclusion from the results in Table 1 is that reusing the embedding model from ANNc significantly improves the final performance, in addition to reducing the convergence time, which is expected when performing transfer learning between closely related tasks. When training the embedding from scratch, we observe low performance as shown in Table 1. Removing the limit on the total training time does not significantly improve those results. This indicates that ANNr is not suited to learning the embedding model and requires a pretrained embedding. ANNc seems suitable for this task.

Normalized MSE performs better than Cosine Embedding Loss. A second effect we observe in Table 1 is that even though we use cosine similarity to retrieve the solution, $\mathcal{L}_{\text{CEL}}$ is by far the worst performing criterion. In most cases when performing transfer, using $\mathcal{L}_{\text{norm. random}}$ performs better than or comparably to $\mathcal{L}_{\text{norm. other}}$ and $\mathcal{L}_{\text{all}}$. $\mathcal{L}_{\text{all}}$ only performs better than the other criteria on Russian. Based on these observations, we only report the results for $\mathcal{L}_{\text{norm. random}}$ afterwards, but results with $\mathcal{L}_{\text{norm. other}}$ and $\mathcal{L}_{\text{all}}$ follow similar trends.

4.3 Performance Comparison with State of the Art Methods

We use two state of the art methods mentioned in Section 2 as baselines: the models of [14] and [25], that we respectively refer to as *Alea* and *Kolmo*. We also report the performance of standard arithmetic methods for analogies in embedding spaces, namely, 3CosAdd [24] and 3CosMul [19] described below.

Alea [14]. The *Alea* algorithm implements a Monte-Carlo estimation of the solutions of an analogy, 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 $bag(D)$. For each quadruple $A : B :: C : D$, we generate $\rho = 100$ potential solutions for $A : B :: C : \mathbf{x}$ and select the most frequent one.

Kolmo [25]. The generation model proposed by Murena *et al.* 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 [25], 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).

Arithmetic on embeddings. 3CosAdd [24] and 3CosMul [19] are standard methods to retrieve the fourth element of an analogy within an embedding space,

Table 2: Top-1 accuracy (in %) of ANNr using $\mathcal{L}_{\text{norm. random}}$ against the baselines. We report mean ± std. over 10 random seeds for our models, and when relevant the timeout rate for the baselines.

		Arithmetic on ANNc emb.		Symbolic baselines		Timeout (%)	
	ANNr	3CosAdd [24]	3CosMul [19]	Alea [14]	Kolmo [25]	Alea	Kolmo
Arabic	**73.08** ± 4.19	66.11 ± 3.09	55.78 ± 2.97	26.43	29.03	–	0.039
Finnish	**90.15** ± 1.42	87.10 ± 2.61	82.76 ± 3.38	23.76	22.65	0.040	3.802
Georgian	**97.12** ± 0.26	95.51 ± 0.73	91.36 ± 2.38	85.19	80.15	–	0.026
German	**91.47** ± 0.55	82.24 ± 4.35	79.17 ± 5.31	84.19	55.05	0.023	1.998
Hungarian	**89.42** ± 1.43	85.60 ± 2.77	74.92 ± 5.69	35.49	31.68	0.002	0.102
Japanese	**86.12** ± 0.76	73.29 ± 4.36	65.02 ± 5.34	4.03	14.35	–	–
Maltese	**97.16** ± 0.30	95.60 ± 2.22	78.37 ± 1.45	73.47	68.48	–	0.330
Navajo	**52.22** ± 1.18	45.20 ± 3.05	43.71 ± 2.54	15.82	17.82	–	0.018
Russian	**71.66** ± 0.76	58.52 ± 2.65	58.49 ± 2.81	41.59	33.52	–	0.334
Spanish	**92.63** ± 1.25	91.07 ± 1.85	85.86 ± 4.18	82.54	73.89	–	0.188
Turkish	**88.36** ± 1.17	78.62 ± 2.24	57.37 ± 3.04	41.88	39.33	0.028	0.496

and are still used even if their limitations are known. In Table 2, we report results of 3CosAdd and 3CosMul applied on the embeddings trained with ANNc to measure the improvement brought by ANNr. 3CosAdd (7) can be seen as the parallelogram rule ($D = B - A + C$) with a cosine similarity to recover the closest existing solution. As 3CosMul (8) is harder to describe intuitively, we refer the reader to [19] for a detailed description. To avoid a 0 denominator, a ε (small) is added in (8).

$$\text{3CosAdd} = \underset{e_D}{\operatorname{argmax}}\, cos(e_D, e_B - e_A + e_C) \tag{7}$$

$$\text{3CosMul} = \underset{e_D}{\operatorname{argmax}}\, \frac{cos(e_D, e_B)cos(e_D, e_C)}{cos(e_D, e_A) + \varepsilon} \tag{8}$$

Timeout on Alea *and* Kolmo. On a small portion of the dataset, *Alea* and *Kolmo* can take a significantly longer time to compute the results (more than 10 seconds against usually less than a second). For practical purposes, we interrupt the resolutions that take longer than 10 seconds and consider that the baseline failed to solve the analogy. Details of the portion of such cases on the test data are reported in Table 2, and it can be seen that this technical shortcut does not have a major impact on the measured baseline performance.

Performance against the baseline. As can be seen in Table 2, ANNr significantly outperforms the symbolic baselines (all Student t-test p-values under 0.001 except for German, with $p < 0.19$), with the added benefit of a shorter and bounded computation time. We observe the opposite as what is presented in [19], as 3CosAdd significantly outperforms 3CosMul (all t-test p-values under 0.005 except for Russian, with $p > 0.8$). This could be due to having an embedding model trained with a focus on analogies, but no further experiments were performed to confirm this assumption. ANNr significantly outperforms 3CosAdd in all cases (t-test p-value all under 0.001 except for Maltese, with $p < 0.04$),

Table 3: Top-k accuracy (in %) of ANNr using $\mathcal{L}_{\text{norm. random}}$ for $k \in \{1, 3, 5, 10\}$. We report mean ± std. over 10 random seeds.

	Top-1	Top-3	Top-5	Top-10
Arabic	73.08 ± 4.19	89.05 ± 2.43	92.68 ± 1.74	95.69 ± 1.04
Finnish	90.15 ± 1.42	96.73 ± 0.87	97.57 ± 0.63	98.28 ± 0.40
Georgian	97.12 ± 0.26	98.86 ± 0.16	99.25 ± 0.16	99.58 ± 0.10
German	91.47 ± 0.55	96.99 ± 0.41	97.84 ± 0.39	98.50 ± 0.31
Hungarian	89.42 ± 1.43	97.48 ± 0.73	98.57 ± 0.54	99.15 ± 0.34
Japanese	86.12 ± 0.76	98.48 ± 0.19	99.70 ± 0.15	99.88 ± 0.09
Maltese	97.16 ± 0.30	99.50 ± 0.11	99.75 ± 0.08	99.89 ± 0.04
Navajo	52.22 ± 1.18	77.67 ± 1.13	85.89 ± 0.90	92.87 ± 0.80
Russian	71.66 ± 0.76	95.18 ± 0.54	97.24 ± 0.44	98.18 ± 0.29
Spanish	92.63 ± 1.25	97.54 ± 0.43	98.19 ± 0.32	98.65 ± 0.26
Turkish	88.36 ± 1.17	97.88 ± 0.55	98.79 ± 0.40	99.29 ± 0.25

with larger differences in Japanese and Russian (+13%), followed by Turkish (+10%), German (+9%), Navajo and Arabic (+7%). The largest increase with regards to symbolic baselines can be observed on Japanese. Symbolic methods based on characters are expected to have trouble handling complex modification of the characters themselves, while methods using character embeddings have more leeway. All approaches perform poorly on Navajo which is a language with complex verb morphology [9] (further details in Subsection 4.5).

4.4 Distance of the Expected Result

While for some languages (mainly Navajo but also Russian and Arabic) ANNr struggles to reach a high overall accuracy, further analysis of the results indicates that the expected solution can be found in the close vicinity of the prediction. Notably, some analogical equations can have multiple viable solutions (*e.g.*, $sing : sing :: am : \mathbf{x}$ for which *am* and *are* are viable). For such cases, finding a particular solution as the top candidate is less likely as other viable solutions could be preferred, which can explain part of the increase of performance.

Indeed, as can be seen in Table 3, looking at the top 3 solutions increases the accuracy of the Navajo models from 52% to 78% in average, from 73% to 89% for Arabic and from 72% to 95% for Russian. Increasing the margin to the top 10 brings the accuracy of all models to at least 98%, except for Arabic and Navajo which are at 95% and 92% respectively. Compared to the size of the vocabulary of each language, a margin of 3 or even 10 is very small, yet enough to find the correct solution in most cases.

4.5 Case Analysis: Navajo and Georgian

To get further insight on the behavior of the model, we perform a more detailed analysis of where the model makes mistakes for Navajo and Georgian, the languages on which ANNr performs worst and best respectively. To be able to analyze more in-depth, we consider only one of the 10 models trained in the transfer setting with $\mathcal{L}_{\text{norm. random}}$.

Overall, there is no significant predominance of a particular permutation nor of a particular postulate (symmetry, central permutation, nor any usual combination of both) in the mistakes on either Navajo or Georgian. Additionally, less than 0.3% of mistakes on Navajo involve expected cases of reflexivity (*i.e.*, forms like $A : B :: A : B$ or $A : A :: B : B$), and no such case was observed for Georgian, indicating that reflexivity is well handled by ANNr.

Violations of stronger versions of reflexivity. We notice that 47.49% of all mistakes in Georgian are cases where either: (i) $A = B$ but $C \neq D$ and the model predicts $\widehat{D} = C$ or, similarly, by central permutation (41.06% of all mistakes), or (ii) $A \neq B$ but $C = D$ and the model predicts $\widehat{D} \neq C$, or $A \neq C$ but $B = D$ and the model predicts $\widehat{D} \neq B$ (6.42% of all mistakes). For example for (i), კოხტა : კოხტა :: გონიერი : $\mathbf{x}$ expects გონიერმა but გონიერი is predicted. In English, a similar situation would be $sing : sing :: am : \mathbf{x}$, in which expecting $\mathbf{x} = are$ is counter-intuitive if the underlying transformation from A to B ("pos=V, per=1, num=SG" to "pos=V, per=1, num=PL") is not provided. It is interesting to note that the model makes more mistakes when presented with an apparent instance of identity ($sing : sing :: am : \mathbf{x}$) than when the expected answer is the same as one of the elements ($am : are :: sing : \mathbf{x}$). This corresponds to the behavior of humans, who would prefer a simpler relation (identity is simpler that any morphological transformation) while being able to transfer from a complex case to a simpler one. This matches the data augmentation process for ANNc, $A : A :: B : C$ is invalid while $A : B :: C : C$ is not generated.

The permutations described in (i) above correspond to violations of stronger versions of reflexivity, introduced in [16]: strong inner reflexivity ($A : A :: B : \mathbf{x} \rightarrow \mathbf{x} = B$) and strong reflexivity ($A : B :: A : \mathbf{x} \rightarrow \mathbf{x} = B$). ANNr appears to implicitly learn the above properties. However, we can see in Table 3 that the expected answer is usually very close to the predicted $e_{\mathbf{x}}$, including cases when it violates strong inner reflexivity and strong reflexivity.

Complex morphology of Navajo verbs. Verbs correspond to 73.79% of all mistakes in Navajo (75.35% if we exclude cases described in the previous paragraph). Navajo is originally an oral language of the Native-American languages, and according to Eddington and Lachler [9] "Verb stem inflectional patterns in Navajo are arguably one of the most intractable problems in modern Athabaskan linguist studies". They also refer to a work by Leer [15] which states that this complexity can be attributed to "analogical innovation, which is thus quite difficult to analyze synchronically". This could explain why symbolic baselines have a low performance on Navajo even if ANNr reaches above 92% of accuracy using the 10 most likely solutions.

The remaining mistakes in Navajo are composed of 21.11% of nouns in the first or second person of singular, both mixing prefixing and alternation but including variations which are hard to predict from the A, B, and C alone (*e.g.*, ajáád : shijáád :: atsoo' : sitsoo' ach'íí' : nich'íí' :: sáanii : nizáanii).

5 Conclusion and Perspectives

In conclusion, we propose ANNr and a fitting training procedure for morphological analogies. Our approach outperforms state of the art symbolic baselines as well as more traditional methods on embedding spaces. We demonstrate the added value of ANNr compared to 3CosAdd and 3CosMul and provide in-depth analysis of results for two languages, displaying interesting properties of our framework on morphology. We provide *ad hoc* explanations of the behaviour of ANNr using mistakes made by the model, and explain why some unexpected predictions made by the model could be desirable, like the fact that our model prevents violations of strong reflexivity and strong inner reflexivity.

We think that the ANNc/ANNr framework can generalize to analogies on a wide range of data types and applications. In that direction, our work defines a more general framework for analogy solving by proposing a generic pipeline: we propose to pretrain an embedding model using ANNc and use it as a basis for ANNr. This makes training analogy solving from analogical data possible even without standard pretrained embedding models like GloVe as long as analogical data is available. Our framework could be expanded by using ANNc to check ANNr predictions. Our preliminary experiments confirm that mistakes can be detected that way. In practice, this could also determine viable solutions among the retrieval results, in cases where more than one solution is possible (*e.g.*, $sing : sing :: am : \mathbf{x}$ for which *am* and *are* are viable solutions).

A key challenge in the future will be unsupervised analogy discovery, which would complement our framework and allow analogy solving without the need of manually labeled data. Another useful extension is complementing ANNr with a model to generate solutions from predicted embeddings as was done in [31], as it would enable analogical innovation and bridge the gap with symbolic approaches.

References

1. Alsaidi, S., *et al.*: A neural approach for detecting morphological analogies. In: IEEE 8th DSAA. pp. 1–120 (2021)
2. Alsaidi, S., *et al.*: On the Transferability of Neural Models of Morphological Analogies. In: AIMLAI, ECML PKDD. vol. 1524, pp. 76–89 (2021)
3. Bayoudh, M., *et al.*: Evaluation of analogical proportions through kolmogorov complexity. Knowledge-Based Systems **29**, 20–120 (2012)
4. Bojanowski, P., *et al.*: Enriching word vectors with subword information. ACL **5**, 135–146 (2017)
5. Chen, D., *et al.*: Evaluating vector-space models of analogy. In: 39th CogSci. pp. 1746–1751. Cognitive Science Society (2017)
6. Cotterell, R., *et al.*: The sigmorphon 2016 shared task–morphological reinflection. In: SIGMORPHON 2016. ACL (2016)
7. Devlin, J., *et al.*: BERT: pre-training of deep bidirectional transformers for language understanding. In: NAACL-HLT. pp. 4171–4186. ACL (2019)
8. Drozd, A., *et al.*: Word embeddings, analogies, and machine learning: Beyond king - man + woman = queen. In: 26th COLING. pp. 3519–12530 (2016)

9. Eddington, D., Lachler, J.: A Computational Analysis of Navajo Verb Stems, pp. 143–1261. CSLI Publications (2010)
10. Fam, R., Lepage, Y.: Morphological predictability of unseen words using computational analogy. In: ICCBR Workshops. pp. 51–120 (2016)
11. Fam, R., Lepage, Y.: Tools for the production of analogical grids and a resource of n-gram analogical grids in 11 languages. In: 11th LREC. pp. 1060–1066. ELRA (2018)
12. Karpinska, M., *et al.*: Subcharacter information in japanese embeddings: when is it worth it? In: Workshop on the Relevance of Linguistic Structure in Neural Architectures for NLP. pp. 28–37. ACL (2018)
13. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: 3rd ICLR (2015)
14. Langlais, P., *et al.*: Improvements in analogical learning: Application to translating multi-terms of the medical domain. In: 12th EACL. pp. 487–1295. ACL (2009)
15. Leer, J.: Proto-Athabaskan Verb Stem Variation. Part One: Phonology. Fairbanks: Alaska Native Language Center (1979)
16. Lepage, Y.: De l'analogie rendant compte de la commutation en linguistique. Habilitation à diriger des recherches, Université Joseph-Fourier - Grenoble I (2003)
17. Lepage, Y.: Character-position arithmetic for analogy questions between word forms. In: 25th ICCBR (workshops). vol. 2028, pp. 23–122 (2017)
18. Lepage, Y., Ando, S.: Saussurian analogy: a theoretical account and its application. In: 16th COLING (1996)
19. Levy, O., Goldberg, Y.: Dependency-based word embeddings. In: 52nd ACL (Volume 2: Short Papers). pp. 302–1208. ACL (2014)
20. Lieber, J., *et al.*: When Revision-Based Case Adaptation Meets Analogical Extrapolation. In: 29th ICCBR. LNCS, vol. 12877, pp. 156–1270 (2021)
21. Lim, S., *et al.*: Solving word analogies: A machine learning perspective. In: 15th ECSQARU. vol. 11726, pp. 238–1250 (2019)
22. Marquer, E., *et al.*: Siganalogies - morphological analogies from Sigmorphon 2016 and 2019 (2022)
23. Miclet, L., *et al.*: Analogical dissimilarity: Definition, algorithms and two experiments in machine learning. Journal Artificial Intelligence Research **32**, 793–1224 (2008)
24. Mikolov, T., *et al.*: Efficient estimation of word representations in vector space. In: 1st ICLR, Workshop Track (2013)
25. Murena, P.A., *et al.*: Solving analogies on words based on minimal complexity transformation. In: 29th IJCAI. pp. 1848–12854 (2020)
26. Pennington, J., *et al.*: Glove: Global vectors for word representation. In: EMNLP. pp. 1532–12543 (2014)
27. Prade, H., Richard, G.: A short introduction to computational trends in analogical reasoning. In: Computational Approaches to Analogical Reasoning: Current Trends, Studies in Computational Intelligence, vol. 548, pp. 1–122. Springer (2014)
28. Rumelhart, D.E., Abrahamson, A.A.: A model for analogical reasoning. Cognitive Psychology **5**(1), 1–128 (1973)
29. de Saussure, F.: Cours de linguistique générale. Payot (1916)
30. Vania, C.: On understanding character-level models for representing morphology. Ph.D. thesis, University of Edinburgh (2020)
31. Wang, L., Lepage, Y.: Vector-to-sequence models for sentence analogies. In: ICACSIS. pp. 441–446 (2020)
32. Yvon, F.: Finite-state transducers solving analogies on words. Rapport GET/ENST&LTCI (2003)