TOWARDS AN OMNILINGUAL NEURAL MODEL FOR SOLVING MORPHOLOGICAL ANALOGIES

Safa AlSaidi
IDMC
Université de Lorraine
Nancy, FR
safa.alsaidi2@etu.univ-lorraine.fr

Amandine Decker
IDMC
Université de Lorraine
Nancy, FR
amandine.decker1@etu.univ-lorraine.fr

Stéphanie Monteiro
IDMC
Université de Lorraine
Nancy, FR
stephanie.monteiro7@etu.univ-lorraine.fr

February 7, 2022

ABSTRACT

In AI and natural language processing (NLP), analogical proportions are defined as statements that take the form "A is to B as C is to D" .They are used in several reasoning and classification tasks. Several analogy-based approaches exist, some of which include semantic and morphological approaches. Other approaches (e.g. axiomatic approach) were used in detecting analogical proportions between character strings. In this paper, we use deep learning approaches to solve morphological analogies. The presented results demonstrate that our approach is competitive compared to the state-of-the art approaches and to our previously obtained results. We also explore the capacity of solving analogies across languages by building monolingual and bilingual analogies.

Keywords Analogy solving · Neural networks · Regression · Morphological word embeddings.

Introduction

Analogy consist of four objects or words A, B, C, and D and draws a parallel between the relation between A and B and the one between C and D. It can be used as a method of reasoning and can be expressed by an analogical proportion, which is a statement meaning "A is to B as C is to D".

Analogies have been extensively studied in NLP and applied to various domains such as derivational morphology [18] as in the following example: "*happy* is to *unhappy* as *friendly* is to *unfriendly*". Building this type of morphological analogies depends on the morphological variations of two words as "happy" and "friendly" in the above-mentioned example. The question of the correctness of an analogy $A : B :: C : D$ is a difficult task; however, it has been tackled both formally and empirically [18, 15, 12]. Recent empirical works propose data-oriented strategies to learn the correctness of analogies from past observations. These strategies are based on machine learning approaches.

An analogical proportion becomes an equation if one of its four objects is unknown [17] as in the following example:

$$R = \{x \,|\, \text{cat is to dog as cats is to } x\}.$$

To solve this form of equations, one needs to calculate the set R of solutions x which satisfy the analogy [17]. In this case, the solution is based on determining the transformation features involved in the morphological variations of words "cat" and "dog".

Analogical classification and inferences are two main problems worth addressing when dealing with the axiomatic settings of analogies. Multiple attempts have been conducted in terms of formulating and manipulating analogies [13, 12]. Fam and Lepage's create analogical grids by relying on feature vectors to detect analogies that exist within a list of words. As a result, they were able to generate analogies that are made up of more than 4 words. Yvon [24] has reformulated the axioms of Lepage [13] to give a closed form solution. This was calculate by Alea in a Monte-Carlo setting, where character strings A, B and C are randomly sliced and merged to obtain potential solutions to $A : B :: C : x$. An approach carried out by Murena *et al.* (2020) [18] outperformed both Alea and Fam and Lepage's results, where analogical equations $A : B :: C : x$ are solved by finding the x that minimizes the Kolmogorov complexitity.

This project is a continuation of last year's work on "A Neural Approach to Detecting and Solving Morphological Analogies across Languages" [2]. For the first part of our project, we worked on building a decoder that would be able to transform our previous regression results (vectors) into words. In the second part of this project, we explored the transferability of our regression model between languages by building bilingual analogies. Eventually, we built a single omnilingual model that is able to tackle both monolingual and bilingual analogies.

This report is organized as follows. We start by defining analogical proportions and explaining how they are solved. We also detail our motivations. Section 2 is devoted to the datasets we used. Our attempts regarding the creation of an embedding decoder as part of a regression task and the challenges we faced are described in Section 3. Section 4 proposes different approaches to deal with transferability, again in a context of solving analogies, but this time with monolingual or bilingual analogies. Finally, the conclusion section outlines the current status of our project and discusses the future perspectives.

To carry out this project, we exclusively wrote our codes in Python (version 3.9) [21]. At first, we used the PyTorch deep learning-oriented library [1] and then we moved to PyTorch Lightning to short the time of our experiments. We also used various Python libraries such that Pandas, Matplotlib, Numpy and Scikit-learn to visualize our data or to access to some support functions. In addition, we worked with Flask to create our software. To conduct the experiments presented in this report, we used computational clusters equipped with GPU from the Grid'5000 testbed [6] (see `https://www.grid5000.fr`). All of our work is available on this GitHub repository: `https://github.com/Safa-98/omnilingual_morphological_analogies`.

1 Background and related works

The importance of analogy in morphology has been long known [7]. The works we mentioned beforehand have paved the way toward the current definition of *proportional analogy* [17, 19] as a 4-ary relation for which the following postulates hold true for all A, B, C, and D:

- $A : B :: A : B$ (reflexivity);
- $A : B :: C : D \rightarrow C : D :: A : B$ (symmetry);
- $A : B :: C : D \rightarrow A : C :: B : D$ (central permutation).

Other equivalent forms can be derived by applying the symmetry and central permutation as such: $D : B :: C : A$, $C : A :: D : B$, $B : A :: D : C$, $D : C :: B : A$ and $B : D :: A : C$. In our project, we focused on *analogy solving*, where we try to finding the values of x such that $A : B :: C : x$ is a valid analogy. In the next subsections, we will recall our last year's obtained results for the regression task.

1.1 Regression

As we mentioned previously, in the regression task we try to solve $A : B :: C : x$ by identifying how B differs from A and generating D that differs in the same way from C. By following *central permutation*, we apply the same operations on $A : C :: B : x$. We use the difference between A and C to obtain D from B. Our regression model (ANNr) architecture is as follows:

- two fully connected layers with ReLU activation on the respective embeddings (A and B for one layer and A and C for the other to determine the relation between A and B and the one between A and C);
- a third fully connected layer without activation generates the embedding of the predicted D from all the extracted information

Due to a mistake in our code, the two fully connected layers going over A and B, and A and C shared the same parameters. After this mistake was discovered, we had two different models: one with shared parameters between the two layers and one with different parameters. We worked with the two architectures to compare their performances, the new results are presented in Section 4.1.1.

We used the datasets described in Section 2 to train the ANNr models. For each of the 11 languages, we trained the model 3 times with different random initialization. Only valid analogies were used to train the models. The obtained results are reported in Table 1 and discussed in this section.

The baselines we refer to are the regression models by *alea* [11] and *Kolmo* [18].

Table 1: Accuracy (in %) of 3 runs of the regression model against the best baseline (Student t-test: *** $p < 0.01$; ** $p \in]0.01, 0.05]$; * $p \in]0.05, 0.1]$).

Language	**ANNr (ours)** (mean ± std.)	**Best baseline**	**T-test**
Arabic	$\mathbf{77.97} \pm 16.03$	28.94 (Kolmo)	**
Finnish	$\mathbf{37.78} \pm 9.28$	22.82 (Kolmo)	*
Georgian	$\mathbf{94.66} \pm 1.13$	86.81 (alea)	
German	$\mathbf{86.38} \pm 0.45$	85.46 (alea)	*
Hungarian	$\mathbf{53.83} \pm 3.12$	35.79 (alea)	**
Maltese	$\mathbf{75.00} \pm 5.08$	74.49 (alea)	
Navajo	$\mathbf{31.74} \pm 0.90$	17.97 (Kolmo)	***
Russian	$\mathbf{75.15} \pm 0.44$	42.00 (alea)	***
Spanish	$\mathbf{86.27} \pm 0.71$	85.23 (alea)	
Turkish	$\mathbf{61.95} \pm 10.86$	42.53 (alea)	*
Japanese	$\mathbf{61.60} \pm 1.33$	18.62 (Kolmo)	***

As can be seen in Table 1, in most cases our model outperforms the baseline with a more or less significant difference. We also note than there is a gap that is greater than a significant 49% ($p < 0.01$) in some cases in the table.

However, there is still a margin of improvement. Our regression model is not able to generate new words: it outputs a vector and we retrieve the closest word embedding in our dataset to get the corresponding word. This method has several limitations. The only words our model can output are the ones in our dataset. Hence, it is impossible for our model to solve analogies where the result does not belong to our dataset. Moreover, the accuracy of our model is not as reliable as it would be with an infinite set of possible answers. Eventually, the dataset may contain some mistakes. A generative model could potentially help us identifying some confusions between the different features.

1.2 Decoders

A decoder is a neural model which takes a vector as input and generates text as output. The input vectors correspond to the numerical representation of some data. For instance in our case, the vectors are the embeddings of the words. The aim of the decoder would be to produce the word corresponding to the embedding.

Most decoders rely on recurrent neural networks (RNN) as they are able to produce outputs of variable lengths, which is very often useful in generation tasks.

As mentioned before, our regression model does not generate new words from the embeddings. It can only generate words that exist in our vocabulary. Therefore, for this project, we wanted to equip our embedding model with a decoder that would transform the embedded vectors into words and thus allow us to generate new words. We believe that the decoder would further improve our model's performance or at least make it usable in broader settings.

1.3 Transfer

In last year's project, we also explored the transferability of our analogies classification model across languages (see Appendix F). The aim of this experiment was to explore the generalization capabilities of our model and to test how much it depends on the training language. We worked with two main settings: *full transfer* and *partial transfer*. In full transfer, we transferred both the embedding and classification models on one language across different languages. Then in partial transfer, we transferred only the classification model (the data is embedded with the "right" embedding model) across different languages [3].

In this project, we also wanted to explore the transferability when solving analogies. To do that, we built bilingual analogies. The description of our method and results is mentioned in Section 4.

2 Dataset

For this project, we first used the SIGMORPHON 2016 dataset [5] and the JAPANESE BIGGER ANALOGY TEST SET [10]. From these two datasets, we have a total of 11 languages, 10 of which are from the SIGMORPHON 2016 dataset and we added Japanese from the JAPANESE BIGGER ANALOGY TEST SET. In this section, we introduce these two datasets and explain some of the preprocessing conducted before the realization part of our project.

2.1 SIGMORPHON 2016 dataset

The SIGMORPHON 2016 datasets [5] we used contained training, development and test data for all of the languages as presented in Table 2. The 10 languages involved are Spanish, German, Finnish, Russian, Turkish, Georgian, Navajo, Arabic, Hungarian and Maltese. Most of these languages have rich inflections [20]. The dataset includes 3 different subtasks: inflection, reinflection and unlabled reinflection. For our project, we work on the data from the inflection task, which is made up of triples $\langle A, F, B \rangle$ of a source lemma A (ex: "cat"), a set of features F (ex: pos=N,num=PL) and the corresponding inflected form B (ex: "cats"). The forms and lemmas are encoded as simple words while the tags are encoded as morpho-syntactic descriptions (MSD), *e.g.,* the grammatical properties of the words such as their part of speech, case or number (among others).

Figure 1 is an example of triples LEMMA, MSD, TARGET FORM from the German training data:

```
Fremdsprache    pos=N, case=ACC, gen=FEM, num=PL      Fremdsprachen
Absorption      pos=N, case=ACC, gen=FEM, num=PL      Absorptionen
absurd          pos=ADJ, case=DAT, gen=FEM, num=SG    absurder
```

Figure 1: Examples from the German training set of the SIGMORPHON 2016 dataset

As the pairs (*Fremdsprache*, *Fremdsprachen*) and (*Absorption*, *Absorptionen*) share the same features, an analogy *Fremdsprache*:*Fremdsprachen*::*Absorption*:*Absorptionen* would be built when loading the dataset. However, the pair (*absurd*, *absurder*) has different features so the analogies *Fremdsprache*:*Fremdsprachen*::*absurd*:*absurder* or *absurd*:*absurder*::*Absorption*:*Absorptionen* would be invalid.

Table 2: Number of analogies for each language.

Language	**Train**	**Dev**	**Test**
Arabic	373,240	7,671	555,312
Finnish	1,342,639	22,837	4,691,453
Georgian	3,553,763	67,457	8,368,323
German	994,740	17,222	1,480,256
Hungarian	3,280,891	70,565	66,195
Maltese	104,883	3,775	3,707
Navajo	502,637	33,976	4,843
Russian	1,965,533	32,214	6,421,514
Spanish	1,425,838	25,590	4,794,504
Turkish	606,873	11,518	11,360

2.2 Japanese Bigger Analogy Test Set

We also used the dataset by Karpinska et al. [10] to produce analogies from pairs of words of the dataset. This dataset contains several files, each of them containing pairs of linguistically related words. As mentioned previously, we are interested in the inflectional and derivational morphology relations. The dataset contains respectively 515 and 502 pairs of words. For each two pairs with the same relation, we produced an analogy, which gave us 26410 analogies in the end, which is much smaller than the SIGMORPHON 2016 one. We split the dataset to get 70 percent for training and 30 percent for testing.

3 Filling the Gap: Decoding of Word Embeddings for Generation of Coherent New Words

Our project is a continuation of our previous work which achieved new state-of-the-art results in both the detection and solving of morphological analogies [4]. We focus on analogy solving, which consists in finding the value of x so that $A : B :: C : x$ is a valid analogy. As a reminder, an analogy is said to be valid when C and D undergo the same transformation as A and B. Precisely, our objective is to improve our regression task. Indeed, our current output for our regression model is a word embedding. In order to obtain this embedding, we have taken the word corresponding to the closest embedding to the generated vector among a set of known embeddings by computing the Euclidean distance. Therefore, it would be interesting to equip our model with decoder to directly access the words corresponding to the generated embedding and to be able to generate new words that don't exist in our vocabulary. In this section, we describe our different approaches to create a decoder for the embedding space as well as our attempt to apply it to our ready-made regression model.

3.1 Approach

In this part, we discuss the two approaches for decoding word embeddings as part of the regression task. The first consists in using the embeddings generated by our pre-existing model and to train a decoder capable of transforming these vectors into words. The second is to train an auto-encoder in order to learn the words representations and then fine-tune it on the regression task.

3.1.1 Decoder for a pre-existing model

Encoder As a reminder, our project follows on from our previous work dealing with morphological analogies and specifically on classification and regression tasks[4]. Therefore, we use our existing encoder inspired by the model of Vania to capture the morphological aspect of words.

We briefly recall the architecture of this embedding model and the training process. This is a CNN character-level word embedding model. It takes as input what we call a pre-embedding. Each word of length w is transformed into a sequence of integers of length $w + 2$ using a character to integer mapping. Two numerical identifiers are added to indicate the beginning (BOS_ID) and the end (EOS_ID) of the word. Then, we convert this list to a PyTorch tensor. Each character of the tensor becomes a learnable vector representation of size m when passing through the character embedding layer. We obtain a matrix of size $w + 2 \times m$ for a word of length w. In the convolutional layer, a total of 80 filters are applied. At the end, we obtain as output a word embedding composed of 80 components using max-pooling.

We trained one encoder for each of the 11 languages coming from SIGMORPHON 2016 dataset [5] and JAPANESE BIGGER ANALOGY TEST SET [10]. We also trained an omnilingual encoder on all the languages apart from Japanese. We trained each of these models together with a classifier model. The aim of the classifier is to determine whether an analogy $A : B :: C : D$ is valid, it thus need morphological clues on the words. Training an encoder together with a classifier should thus force the encoder to learn useful morphological clues.

Decoder The model described in this paragraph follows the architecture of the decoder proposed in [8] and is illustrated in Figure B.1. As the training and evaluation phases differ, we explain these two steps separately.

Training The dataloader gives three different kinds of representation of a word:

- a *word embedding* which is a tensor of length 80;
- a *packed input* which corresponds to a representation of the word in IDs from the BOS_ID to the last letter ID. eg. [41, 23, 15, 18, 4] with {4: 'd', 15: 'o', 18: 'r', 23: 'w', [...], 41: 'BOS', 42: 'EOS'} as mapping dictionary
- a *packed output* which refers to a representation of the word in IDs from the first letter ID to the EOS_ID. eg. [23, 25, 18, 4, 42]

Our decoder is a GRU-based deep learning model. It takes 2 different inputs:

- an input hidden state: In our case, we use the word embedding and reduce its dimension from 80 to 64 thanks to a linear layer and an activation function;
- a cell input: For the first cell, the input is the embedding of the first character id (here, BOS_ID) in the packed input of the word. For the next cells, the inputs are the embeddings of the next character IDs. Overall it corresponds to the content of the packed input mentioned before. Usually with recurrent neural networks, the output of the previous step is used to build the new input. However in our case, we used the expected output. This method is called *teacher forcing*, it enables the model to be trained faster but it can make it less robust [9].

For each inputted character, the model outputs a distribution of probabilities over the vocabulary. We take the most probable character at each step to build the complete output. Since we use teacher forcing, the model generates the exact number of characters in total[1]. We can thus compare the output with the expected packed output mentioned before. We use cross entropy loss to perform this comparison.

Evaluation In the evaluation step, we know only the first character to decode: BOS. We thus use the word embedding and the BOS_ID as inputs of the model. It will then output a distribution of probabilities over the vocabulary and the decoded character will be the one with the highest probability. We use this character as next input and repeat the process until the EOS_ID is decoded (or when 500 characters have been decoded[2].

We get a sequence of character IDs as output, so we map them to the corresponding characters to get the decoded word.

We decided to use two metrics to evaluate our decoder:

- The accuracy to get the decoder success rate, *i.e.,* the portion of decoded words exactly matching the inputs;
- The Levenshtein distance [14], it gives a measure of the difference between two strings. It is equal to the minimum number of characters that must be deleted, inserted or replaced to switch from one string to another. So, the idea is to compute the distance between the input word of the encoder and the decoded word and expect a score of 0. A Levenshtein distance equal to 0 means that there is no transformation to be done to pass from the input word to the output word (*i.e.*, input word = output word), which would mean that the decoder is efficient.

For the evaluation, we also carried out a small online questionnaire survey. We looked for native or fluent speakers in our eleven languages. We built one file per language where the outputs of our decoder and the target words are stored. We asked the participants to answer three main questions:

- Provide a description of the errors encountered in terms of morphology
- How much of the words formed by the decoder are not real?
- How close is the produced word to the target word on a scale?

We chose to train decoders based on our embedding models from last year because we had good results for the classification task and promising ones for the solving task. However, it requires data in the form of analogies, which is not the most easily accessible for some languages. We would not need this type of data to train an auto-encoder: simple words would be used. This is the reason why we chose to train auto-encoders then.

3.1.2 Auto-encoder

Our second approach was to directly train an auto-encoder. In this case, we trained the encoder and the decoder together without any auxiliary task. We used the same encoder and decoder as in the first presented approach. The auto-encoder takes as input a word and the goal is to obtain the same word as output. The intermediary representation corresponds to the word embedding.

Our encoder part is still a CNN based model with filters of size 2 to 6. Our decoder is composed of 1 GRU layer where the hidden layer input is the word embedding and the cells input is the last decoded character's ID which start with BOS_ID. We also used the same evaluation metrics as for the decoder.

However the results of the evaluation didn't prove that the learned representation contained morphological clues. We thus trained classification models based on the auto-encoder. The results described in Figure E.5 showed us that the embeddings contained enough morphological clues to classify analogies. These results encouraged us to pursue with the analogy solving task.

3.1.3 Regression model

The next step was to train our analogy solving model [4] based on the auto-encoder. We briefly present the architecture of the analogy solving model in Section 1.1 and the auto-encoder in the above section.

[1]If we had not used teacher forcing, we would have kept generating characters until the EOS_ID is decoded.
[2]All the words of our vocabulary are shorter than 500 characters.

3.2 Results and discussion

In this part, we will discuss the results we got for both the decoder and the auto-encoder and then those when we applied the auto-encoder on the analogy solving task.

3.2.1 Decoder & Auto-encoder

In the architecture of our decoder, the word embedding goes through a linear layer followed by an activation function before it is used as input for the GRU. We tested our model with several activation functions: relu, tanh and sigmoid. Moreover, the experiments showed that increasing the hidden size of the GRU increases the accuracy but the increase starts to stop at 1024 or 2048. Figure C.2 illustrates these results with Navajo. The results for the other languages were similar.

The best parameters for our decoder are the sigmoid activation function and an hidden size of 2048. The results vary by language. Our decoder performed well with Navajo, Hungarian, Turkish and Arabic as evidenced by the high accuracy and the low levenshtein distance in Appendix C. We can also notice that the accuracy is really poor regarding Japanese and Finnish. For these two languages, the levenshtein distance indicates that the output is very far from the target.

Regarding the manual evaluation, we obtained some leads on what was wrong with our decoder thanks to our participants. They pointed out that prefixes and suffixes do exist in the languages, which is why the produced words seem to exist in the different languages. The portion of real words varies by language: around 65% for German, none for Russian, 15% for Spanish and 10% for Arabic. These results gave us some leads to investigate such as the way the model decodes the central part of a word. The decoder seem to prioritize intern coherence over the content of the embedding. We tried some minor modifications later on to solve this issue but as explained in Section 3.2.2, the results were not the expected ones. Given this, the idea was to force the model to work with embeddings which seem to prioritize intern coherence.

For the auto-encoder aspect, the accuracy results agree fairly well with those of the decoder (see Figures C.3 and D.4). However, this time, the standard deviation is very large. Moreover, for all languages, the levenshtein distance has inflated which means that there is a big difference between the output word and the input word. It is also due to the large number of outputs of length 500, the maximal length we allowed. Such outputs are usually either completely random or accurate for the first characters but followed by the indefinite repetition of the last characters.

Thus, the results suggest that the decoder is more stable than the auto-encoder. It may be due to the fact that the auto-encoder has no auxiliary task to help focusing on the important aspects of the words. In contrast, the embeddings used as inputs of the decoder already contain the meaningful clues. Moreover, training an auto-encoder is less constraining than training an encoder and a decoder separately. Therefore, we chose to keep working with auto-encoders but we fine-tuned them with the analogy solving task. However, we decided to start experimenting with only two of the eleven available languages. It reduced the need in computational resources and made it easier for us to analyse the results. Indeed, we chose to keep working with German and Arabic, which are two languages one of us knew.

3.2.2 Analogy solving

When applying the auto-encoder on the analogy solving task, we did not get satisfactory results for the two languages on which we tested it. Indeed, we initially worked with Arabic and German to verify the potential of the regression model. We chose these two languages as we are familiar with them and therefore the output results of the experiments we conducted were easier to analyze. However, none of the decoded words were the ones we expected. All the outputs were real words but they were very redundant: the model produced about 30 different words in German and 22 in Arabic. So, our model seemed to be stuck on few words. We tried some modifications to solve this problem such as:

- Using a bidirectional GRU layer, but the model returned empty words;
- Concatenating the embeddings to the input cells, but we obtained random outputs of 500 characters, which is the maximal length we allowed.

One solution would be to use a Variational Auto-Encoder (VAE) to counter this common issue of decoders. Unfortunately, time was not in our favor and forced us to stop there concerning the decoder. Plus, the results of our first trials was very discouraging.

4 Building Multilingual Analogies

Given our poor results with the decoder and auto-encoder, we headed for a mitigation plan that involves solving bilingual morphological analogies based on transfer learning. Our analogical equations still consist of four words A, B,

C, and D but this time, A and B are in a language 1, and C and D in a language 2. The language 1 is called the source language and the language 2, the target language. The objective is to find D knowing that A and B must differ in the same way as C and D.

4.1 Approach

In this part, we will discuss different approaches to tackle bilingual analogies and their respective results.

4.1.1 Regression model

Our previous regression model, described in Section 1.1, outperformed the state-of-the-art results. To our pleasant surprise, we obtained even better results than the old ones for some languages when retraining our monolingual models (see Table 3 below). As mentioned previously, we now have two different analogy solving models. One with shared parameters between the two first fully-connected layers (named *shared parameters*) and one with different parameters (named *diff. parameters*).

There is no significant difference between the results of our two models. However, compared to the previous results, the accuracy has significantly augmented for most languages. For Finnish and Hungarian in particular, the accuracy indicates a considerable improvement. For Georgian and Russian however, the results have slightly decreased. We can note that they remain nonetheless close to the previous results.

Table 3: Accuracy (in %) of the regression models (10 runs for the new results, 3 for previous ones)

Language	**ANNr (previous)** (mean ± std.)	**shared parameters** (mean ± std.)	**diff. parameters** (mean ± std.)
Arabic	77.97 ± 16.03	59.14 ± 1.76	61.13 ± 0.83
Finnish	37.78 ± 9.28	**76.61 ± 1.15**	**76.46 ± 1.58**
Georgian	**94.66 ± 1.13**	85.51 ± 2.00	84.67 ± 2.78
German	86.38 ± 0.45	**89.26 ± 0.51**	**88.70 ± 0.58**
Hungarian	53.83 ± 3.12	**78.49 ± 0.65**	**78.72 ± 0.53**
Maltese	75.00 ± 5.08	77.37 ± 2.27	78.04 ± 1.44
Navajo	31.74 ± 0.90	**46.14 ± 0.54**	**45.74 ± 0.99**
Russian	**75.15 ± 0.44**	72.51 ± 0.46	72.23 ± 0.44
Spanish	86.27 ± 0.71	**91.18 ± 0.51**	**91.72 ± 0.43**
Turkish	61.95 ± 10.86	**80.37 ± 0.82**	**80.37 ± 1.00**
Japanese	61.60 ± 1.33	**74.75 ± 1.33**	**72.58 ± 2.47**

After these first monolingual experiments, we adapted our methodology to our new task: solving bilingual analogies. We first selected three pairs of languages only to see if this idea was feasible. Our choice was based on two criteria:

- Both languages must have the same transformations (*i.e.*, same features).
- The two languages must transfer well from one to another based on the classification results from last year (see Appendix F). Thus, we first excluded the languages with a different alphabet (i.e. Russian and Georgian) because we knew that they don't transfer well. As Arabic is romanized in the dataset, this language was not set aside.

To extract the pairs meeting the first criterion, we implemented a function to compute the number of overlapping features for two languages (see Figure 2 below). We considered that with 50,000 available analogies, a pair of languages could be suitable. It is indeed the number of analogies we used for the monolingual experiments.

Then, among all these possible pairs, we have selected the three pairs with the best transfer accuracy (see Appendix F):

1. Hungarian and German
2. Turkish and Finnish
3. Hungarian and Finnish

For this experiment, we decided to use a single embedding model for the four words. It means that for an experiment on the pair (Hungarian, German), we used the Hungarian embedding model for Hungarian and German words. Only

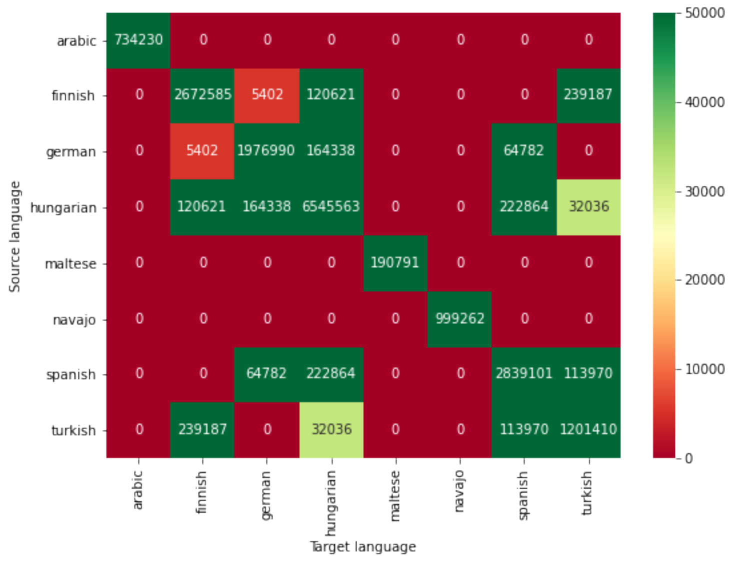


Figure 2: Number of possible analogies per language pair

the analogy solving model was trained during these first trials. The embedding model was frozen in order to see if transferring an encoder was feasible.

Table 4: Accuracy for the analogy solving task on the three language pairs and information about the dataset.

Languages		**Analogy solving accuracy**		**Information on the language pairs**		
Source language	**Target language**	**Cosine similarity**	**Euclidean distance**	**Nb different features**	**Transfer accuracy (%)**	**(source, target) nb of features**
Hungarian	German	58.9	57.7	3	90	(86, 98)
Turkish	Finnish	39.5	39.1	6	81	(187, 95)
Hungarian	Finnish	18.9	16.8	19	84	(86, 95)

The results we obtained for this first trial are described in Table 4. They are poor in comparison to the monolingual ones but we believed we could improve them by fine-tuning the embedding models.

Moreover, the monolingual models were evaluated on the entire dataset, *i.e.*, all the possible features for one language. This was not the case for the bilingual models as only the features shared by the two languages could be used.

Another interesting remarks was that the accuracy of the bilingual models increased when the number of different shared features between the two languages decreased. A possible explanation is that with more different features, the morphological variety in the data most likely increases. In order to verify this experiments but also have a more complete overview of the transferability between the languages, we decided to extend our experiments with other language pairs. We kept all the languages which used the Latin alphabet and had shared features with a least one other such language. Therefore we ended with Finnish, German, Hungarian, Turkish, and Spanish.

4.1.2 Comparison with the monolingual results

As mentioned previously, the bilingual models were evaluated on only a subset of features while the monolingual ones were evaluated on all the features available for one language. In order to compare the performance of the monolingual and bilingual models, we evaluated the monolingual models on several subsets of analogies. For instance, to compare the results of the bilingual model (Hungarian to German), we evaluated the monolingual Hungarian model on the subset of features which are shared with German, *i.e.* the triples colored in gray in Table 5a.

The results we obtained for the bilingual and monolingual models are described in Tables 6a and 6b. We present the results with the analogy solving models where the parameters are different for the two first linear layers (more

Table 5: Example of shared features by Hungarian and German

tekint	pos=V,tense=PRS	tekintő
nagydíj	pos=N,case=ACC,num=PL	nagydíjakat
ural	pos=V,tense=PST	uralt
tákol	pos=V,polite=INFM,per=2,num=SG,finite=NFIN	tákolnod
māhirun	pos=ADJ,def=INDF,case=GEN,gen=MASC,num=DU	māhirayni

(a) Hungarian

plan	pos=ADJ,comp=CMPR	planer
schönen	pos=V,tense=PRS	schönend
Milchprodukt	pos=N,case=ACC,gen=NEUT,num=SG	Milchprodukt
krähen	pos=V,mood=IND,tense=PST,aspect=PFV,per=2,num=SG	krähtest
applaudieren	pos=V,tense=PST	applaudiert

(b) German

explanation on this architecture was given in Section 1.1). We explain in Section 4.1.3 why we chose this architecture over the other. The results with the other architecture are available in Appendix G.2.

Table 6: Accuracy (±std) for the analogy solving task (source languages on rows, target languages in columns). Bold result indicate a significant difference (p-value < 0.05 in the Student t-test)

	Finnish	German	Hungarian	Spanish	Turkish
Finnish	/	39.75±2.67	**80.33±1.95**	/	**79.89±2.43**
German	94.26±0.63	/	**70.55±2.61**	**70.55±2.61**	/
Hungarian	43.93±3.44	**85.39±1.76**	/	**85.39±1.76**	**40.98±3.42**
Spanish	/	**94.26±0.53**	**94.26±0.53**	/	**95.83±0.24**
Turkish	**64.98±2.76**	/	**70.74±2.23**	**94.03±3.70**	/

(a) Monolingual analogies (10 runs)

	Finnish	German	Hungarian	Spanish	Turkish
Finnish	/	66.01±33.01	36.28±0.76	/	28.34±0.48
German	64.43±32.23	/	30.53±0.57	11.92±3.30	/
Hungarian	**50.61±2.26**	77.98±1.24	/	94.02±0.29	33.89±0.77
Spanish	/	32.42±17.01	78.99±0.13	/	40.45±1.52
Turkish	46.43±1.24	/	16.07±0.90	70.86±0.04	/

(b) Bilingual analogies (5 runs)

As seen in Figure 2, Finnish and Spanish, and German and Turkish have no shared features. This is the reason why no result appear in Tables 6a and 6b. Moreover, we have also excluded the results for which the source language and the target language are identical in the monolingual table for reasons of readability and therefore facilitate the comparison with the bilingual table.

For the monolingual results, the accuracy for the pairs (Finnish, German), (Hungarian, Finnish), and (Hungarian, Turkish) is low. We believe that this might be due to some irregularities, invalid analogical forms (analogies of the form A:A::B:C) or more complex transformations. Table 7 shows some Finnish (lemma, target form) pairs with the features pos=ADJ,case=NOM,num=PL. As this transformation is shared with German, analogies built on these pairs of words are in the subset evaluated in the monolingual for the pair (Finnish, German). We can notice that the transformations to go from the lemma to the target are different for each pair. A wide variety of transformations for one set of features may be complicated to learn for the model and explain the poorest results for some pairs.

Table 7: Example of pairs (lemma, target form) for Finnish with the features pos=ADJ,case=NOM,num=PL

Lemma	**Target form**
vaikutusalt**is**	vaikutusalt**tiit**
ronkeli	ronkeli**t**
kuivun**u**t	kuivun**ee**t
kahde**s**toista	kahde**nnet**toista
sulok**as**	sulok**kaat**

For all the pairs except (Hungarian, Finnish), the bilingual results are significantly lower than the monolingual ones. This may be due to the fact that a same transformation does not work the same in two languages as we can see in Table 8. Given the current state of the model and the quality of the results, we realized that retraining the bilingual models would be computationally too costly. Nevertheless, we believe that solving bilingual analogies can have a positive impact for learning for instance. For this reason, we then worked on a single analogy solving model which would be able to solve both monolingual and bilingual analogies.

Table 8: Example of (Finnish, German) analogies

Adjective, comparative			
ajatuksek**as**	ajatuksek**kaampi**	atemberaubend	atemberaubend**er**
Adjective, nominative case, plural			
harmaantun**ut**	harmaantun**eet**	deliziös	deliziös**e**
ronkeli	ronkeli**t**	hasserfüllt	hasserfüllt**e**
Adjective, accusative case, plural			
arma**s**	arma**at**	sauerstoffhaltig	sauerstoffhaltig
terve	terve**et**	gelöst	gelöst

4.1.3 Omilingual model

In the before-experiments, we were working with one model per language, or one model per pair of language. An interesting idea would be to have a single model that would be able to solve analogies in several languages. We thus decided to build an omnilingual analogy solving model. We chose to work with Finnish, German, Hungarian, Turkish, and Spanish again so that our model could solve both monolingual and bilingual analogies.

We already had an omnilingual embedding model from our previous project. This model was trained on the 10 languages of the SIGMORPHON 2016 dataset and is thus suitable for this new experiment.

The first step to build our omnilingual analogy solver model was to merge the datasets of the five languages into one. We simply concatenated the datasets. Therefore when loading the data, we got analogies for all the pairs of languages sharing features (including monolingual analogies as it corresponds to the setting *source language* = *target language*).

The results we obtained with both architectures (shared parameters and different parameters) for the analogy solving model are described in Tables 9a and 9b. Bold results indicate a significant difference for one of the architectures. These results show that we obtained better results when using different parameters for the first two linear layers of the analogy solving model. This may be due to the learning capacity of the model: there are more parameters and thus a greater learning possibility when the two first linear layers are different. It would be interesting to see if we could get similar results with shared parameters between the two first linear layers if these layers were larger.

Table 9: Accuracy (in %) of 10 runs of the omnilingual regression model

	Finnish	German	Hungarian	Spanish	Turkish
Finnish	61.85±1.79	3.22±1.73	28.58±2.69	/	55.23±3.85
German	3.15±1.47	64.38±1.27	**66.91±4.49**	62.07±3.58	/
Hungarian	36.26±4.52	55.25±1.47	**73.33±1.31**	**78.36±1.33**	**32.00±3.03**
Spanish	/	61.16±2.54	**74.05±1.77**	69.38±1.65	70.67±4.03
Turkish	54.12±1.48	/	25.38±3.94	65.72±6.09	**52.23±1.09**

(a) Different parameters

	Finnish	German	Hungarian	Spanish	Turkish
Finnish	60.95±1.75	3.34±1.16	30.34±2.02	/	53.82±2.74
German	3.05±1.43	63.78±0.88	60.99±4.86	62.65±3.47	/
Hungarian	35.06±4.49	56.35±1.96	71.69±1.84	70.61±7.90	27.10±3.17
Spanish	/	61.90±2.42	67.32±5.05	67.90±1.97	66.12±6.86
Turkish	54.00±2.41	/	24.19±2.21	64.94±5.61	50.39±1.64

(b) Shared parameters

The results we obtained with the omnilingual model are overall poorer than the monolingual ones. It performs significantly better than the bilingual model for seven pairs nonetheless (see Table G.5 in Appendix). We believe that the performance is acceptable as we use only one model instead of one per pair of languages. Training the omnilingual embedding model on more data and balancing the number of analogies for each pair of languages in the omnilingual dataset may also improve the results of this model.

4.2 Software

We have created a website using the Python micro-framework Flask, HTML and CSS (see Appendix H). We have designed our site to engage the user in a learning process. We named it 'ANNa' which stands for Analogies and Neural Networks. We used the same name as the website A*NN*a, a platform presenting neural network based tools to work with analogies. The two tools it contains for now are based on the work we did last year, with our supervisors Miguel Couceiro and Esteban Marquer, on classification and solving of morphological analogies.

Our software uses our omnilingual model to solve both monolingual and bilingual analogies. We decided to exploit a sample of the dataset and not the entire dataset to prevent our website to be slow.

The first step consists in selecting a source language and a target language among a list of languages and generate an example of analogy where D is missing. Of course, when the source language is the same as the target language, it generates an example of monolingual analogy and when they are different, an example of bilingual analogy.

Then, the user can either write their own answer or generate the result of our omnilingual model by clicking on the button "Get closest result". This button acts like the famous help button although we are not sure that it returns the correct result. Consequently, another button called "Get the right answer" indicates whether the answer given either by the user or the model is correct or not with a color system. When the response is valid, the box turns green whereas when it is incorrect, the box is red and a message displays the true answer.

Once the analogy is corrected, an option gives the possibility to shuffle the words in the equation while remaining a valid analogy.

We have also introduced an advanced option to choose a transformation. This can be particularly useful when, for example, a user who only knows German wants to know how the plural in Hungarian works. Thus, the user can access examples and draw a parallel between the two languages. For this, we have implemented a function to have a more readable format of the morpho-syntactic descriptions available in the dataset using this glossary [22].

At the top of the page, the user can find tabs which refer to different sections or another page containing a description of ourselves, of the application, explanations about analogies and the morphology of the five languages used. The question mark icons (see Figure H.8) in the application part leads to one particular explanation in the explanation section (see Figure H.9) to ensure and improve the user's understanding.

5 Conclusion and Perspectives

With this project, we tried to improve the neural model for solving morphological analogies we worked on last year. We had two ideas in mind. On the one hand, building a decoder which would be able to produce words based on the embeddings produced by our analogy solving model. This would enable us to generate words when solving analogies instead of retrieving results from our dataset. On the other hand, we wanted to explore more the transferability of our model from one language to another. We showed last year that some embedding models could be transferred to other languages when classifying analogies. Being able to do the same thing when solving analogies could help working with low-resource languages.

Working on the decoder led us to work with recurrent neural networks, which we had never done before. Even if the final results were not the ones we expected, it was an enriching experience. The next step would be to explore more complex decoders and auto-encoders such as variational auto-encoders (VAE). The issues we encountered with our auto-encoder are indeed well known and VAEs are a possible solution to solve them.

The multilingual aspect of our project brought better results. While solving bilingual analogies revealed that transferring one model to another language was a complex task, we managed to build one model to solve analogies in several languages. A model could thus take advantage of learning the features of different languages. The results we obtained with our omnilingual model were still lower than the ones with several monolingual models. However, this model performed significantly better than the bilingual ones for several pairs of languages. We believe that training the model with a more balanced dataset could improve the results significantly.

Moreover, the dataset we worked with contained only 10 languages. Among them, only five were suitable for our experiments as they shared some morphological transformations. However, the number of shared features is significantly smaller than the total number of features of a language. Extending our experiments to other languages, which would share more transformations with each other could help us understand more what our model learns and potentially improve it. We could also explore the generalization capability of our omnilingual model with a larger dataset. Indeed, we could apply our trained models to new data and see if the performance remains the same. The Sigmorphon 2019 [16] seems interesting for these new experiments.

We have in perspective to publish a paper in collaboration with our professors Miguel Couceiro and Esteban Marquer to present our results with the multilingual setting.

6 Acknowledgments

We would like to thank our professors, Miguel Couceiro and Esteban Marquer for their practical guidance, constructive advice and endless support throughout this semester. We also greatly appreciated the participation of our classmates in our survey. Last but not least, we are extremely grateful to all lecturers of IDMC for teaching us to gain a deeper understanding on developing this project. Finally, we are very grateful to IDMC for giving us the chance to carry out a project.

References

[1] A. Paszke *et al.* (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Proceedings of Neurips on Advances in Neural Information Processing Systems 32*, pages 8024–8035.

[2] Alsaidi, S., Decker, A., Lay, P., Marquer, E., Murena, P.-A., and Couceiro, M. (2021a). A Neural Approach for Detecting Morphological Analogies. In *The 8th IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, Porto/Online, Portugal.

[3] Alsaidi, S., Decker, A., Lay, P., Marquer, E., Murena, P.-A., and Couceiro, M. (2021b). On the Transferability of Neural Models of Morphological Analogies. In *AIMLAI, ECML PKDD 2021: European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, Bilbao/Virtual, Spain.

[4] Alsaidi, S., Decker, A., Marquer, E., Murena, P.-A., and Couceiro, M. (2021c). Tackling morphological analogies using deep learning – extended version. 2111.05147.

[5] Cotterell, R., Kirov, C., Sylak-Glassman, J., Yarowsky, D., Eisner, J., and Hulden, M. (2016). The sigmorphon 2016 shared task—morphological reinflection. In *Proceedings of the 2016 Meeting of SIGMORPHON*, Berlin, Germany. Association for Computational Linguistics.

[6] D. Balouek *et al.* (2013). Adding Virtualization Capabilities to the Grid'5000 Testbed. In Ivanov, IvanI., Sinderen, Marten, Leymann, Frank, Shan, and Tony, editors, *Cloud Computing and Services Science*, volume 367 of *Communications in Computer and Information Science*, pages 3–20. Springer International Publishing.

[7] de Saussure, F. (1916). *Cours de linguistique générale*. Payot, Paris.

[8] Dwaraknath, R. V. (2018). Generating words from embeddings.

[9] Goyal, A., Lamb, A., Zhang, Y.-J., Zhang, S., Courville, A. C., and Bengio, Y. (2016). Professor forcing: A new algorithm for training recurrent networks. In *NIPS*.

[10] Karpinska, M., Li, B., Rogers, A., and Drozd, A. (2018). Subcharacter Information in Japanese Embeddings: When Is It Worth It? In *Proceedings of the Workshop on the Relevance of Linguistic Structure in Neural Architectures for NLP*, pages 28–37, Melbourne, Australia. Association for Computational Linguistics.

[11] Langlais, P., Yvon, F., and Zweigenbaum, P. (2009). Improvements in analogical learning: Application to translating multi-terms of the medical domain. In *EACL 2009, 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 487–495.

[12] Lepage, Y. (2003). *De l'analogie rendant compte de la commutation en linguistique*. Habilitation à diriger des recherches, Université Joseph-Fourier - Grenoble I.

[13] Lepage, Y. (2004). Analogy and formal languages. In *Joint meeting of the 6th Conference on Formal Grammar and the 7th Conference on Mathematics of Language*, volume 53, pages 180–191.

[14] Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710. Doklady Akademii Nauk SSSR, V163 No4 845-848 1965.

[15] Lim, S., Prade, H., and Richard, G. (2019). Solving word analogies: A machine learning perspective. In Kern-Isberner, G. and Ognjanovic, Z., editors, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty, 15th European Conference, ECSQARU 2019, Belgrade, Serbia, September 18-20, 2019, Proceedings*, volume 11726 of *Lecture Notes in Computer Science*, pages 238–250. Springer.

[16] McCarthy, A. D., Vylomova, E., Wu, S., Malaviya, C., Wolf-Sonkin, L., Nicolai, G., Kirov, C., Silfverberg, M., Mielke, S. J., Heinz, J., Cotterell, R., and Hulden, M. (2019). The SIGMORPHON 2019 shared task: Morphological analysis in context and cross-lingual transfer for inflection. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 229–244, Florence, Italy. Association for Computational Linguistics.

[17] Miclet, L., Bayoudh, S., and Delhay, A. (2008). Analogical dissimilarity: Definition, algorithms and two experiments in machine learning. *Journal of Artificial Intelligence Research*, 32:793–824.

[18] Murena, P.-A., Al-Ghossein, M., Dessalles, J.-L., and Cornuéjols, A. (2020). Solving analogies on words based on minimal complexity transformation. In Bessiere, C., editor, *29th IJCAI*, pages 1848–1854.

[19] Prade, H. and Richard, G. (2014). A short introduction to computational trends in analogical reasoning. In Prade, H. and Richard, G., editors, *Computational Approaches to Analogical Reasoning: Current Trends*, volume 548 of *Studies in Computational Intelligence*, pages 1–22. Springer.

[20] R. Cotterell *et al.* (2018). The CoNLL–SIGMORPHON 2018 shared task: Universal morphological reinflection. In *the Proceedings of the CoNLL–SIGMORPHON 2018 Shared Task on Universal Morphological Reinflection*, pages 1–27.

[21] Rossum, G. V. and Drake, F. (2009). Python 3 reference manual.

[22] Sylak-Glassman, J. (2016). The composition and use of the universal morphological feature schema (unimorph schema). *Johns Hopkins University*.

[23] Vania, C. (2020). *On understanding character-level models for representing morphology*. PhD thesis, University of Edinburgh.

[24] Yvon, F. (2003). Finite-state transducers solving analogies on words. *Rapport GET/ENST&LTCI*.

Appendices

A Glossary

Table A.1: Acronyms used in the report

Acronym	Full name
BOS	Beginning Of Sequence
CNN	Convolutional Neural Network
EOS	End Of Sequence
GRU	Gated Recurrent Unit
ID	Identifier
MSD	Morpho-syntactic descriptions
NLP	Natural Language processing
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Networks
VAE	Variational Auto-Encoder

B Architecture of our GRU decoder

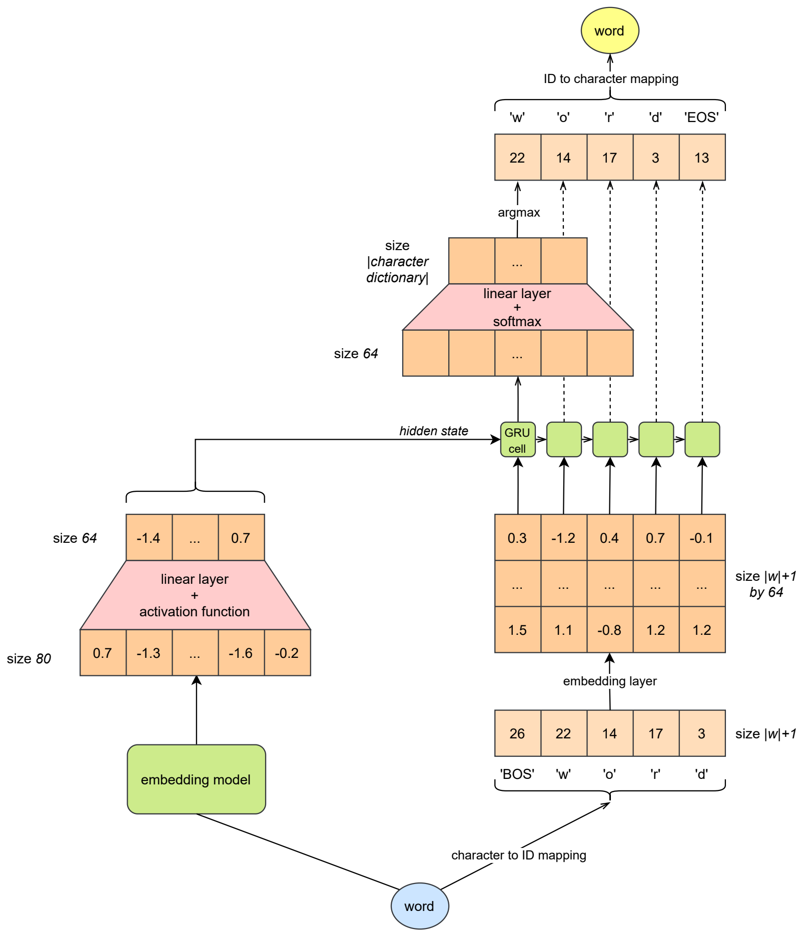


Figure B.1: GRU model

C Decoder results

C.1 Trials with different parameters

We use word embeddings as input of our decoders. Our embedding model produces vectors of size 80. Before going through the GRU model, the embeddings go through a linear layer followed by an activation function. We tried different activation functions. We also tried different sizes for the hidden layer of the GRU model.

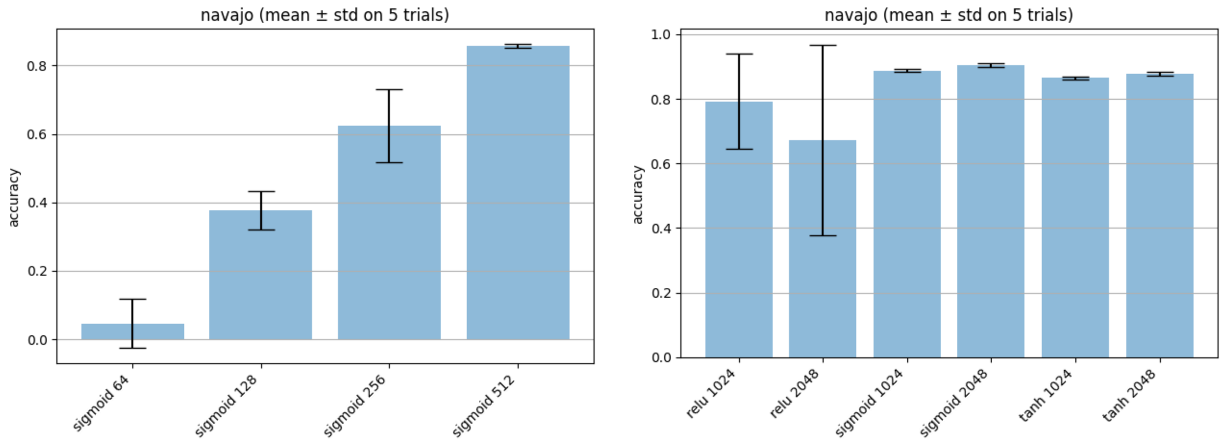
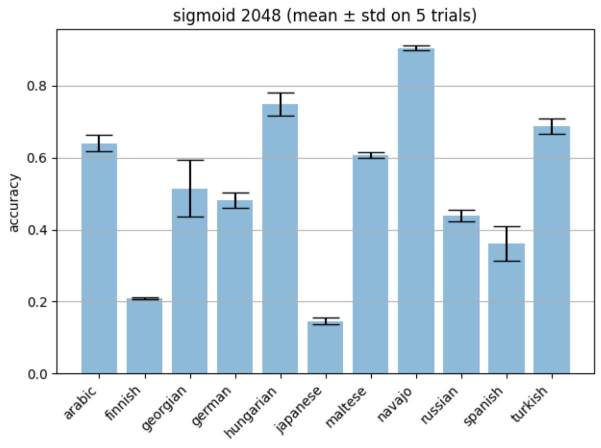
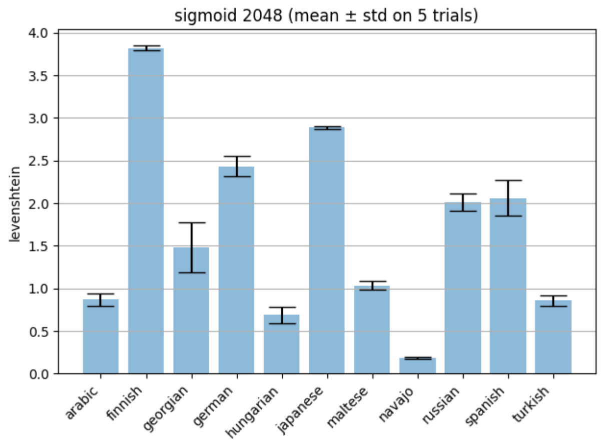


Figure C.2: Accuracy (%) with different parameters

C.2 Results for all the languages



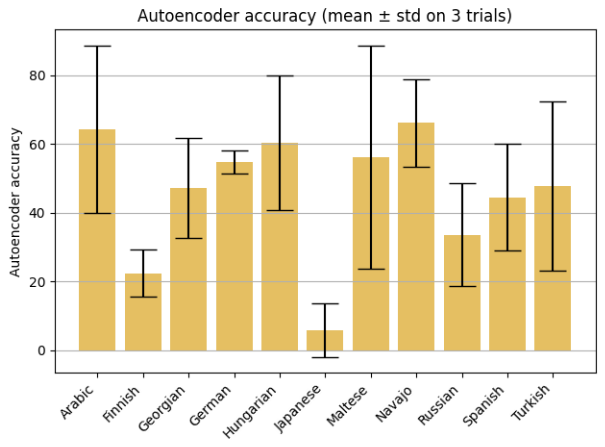
(a) Accuracy



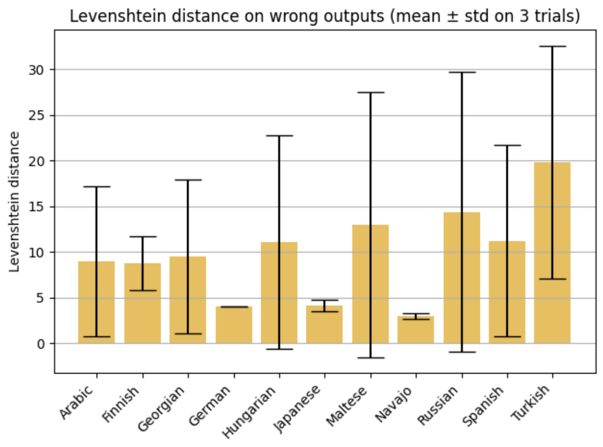
(b) Levenshtein distance

Figure C.3: Decoder results for all the languages

D Auto-encoder results



(a) Accuracy



(b) Levenshtein distance

Figure D.4: Auto-encoder results for all the languages

E Classification results based on the auto-encoder

E.1 Results of last year

Table E.2: Accuracy results (in %) for the classification task.

Language	**Valid analogies**	**Invalid analogies**
Arabic	99.89	97.52
Finnish	99.44	82.62
Georgian	99.83	91.71
German	99.48	89.01
Hungarian	99.99	98.81
Japanese	99.99	98.65
Maltese	99.96	77.83
Navajo	99.53	90.82
Russian	97.95	79.85
Spanish	99.94	78.33
Turkish	99.48	92.63

E.2 Results with the auto-encoder

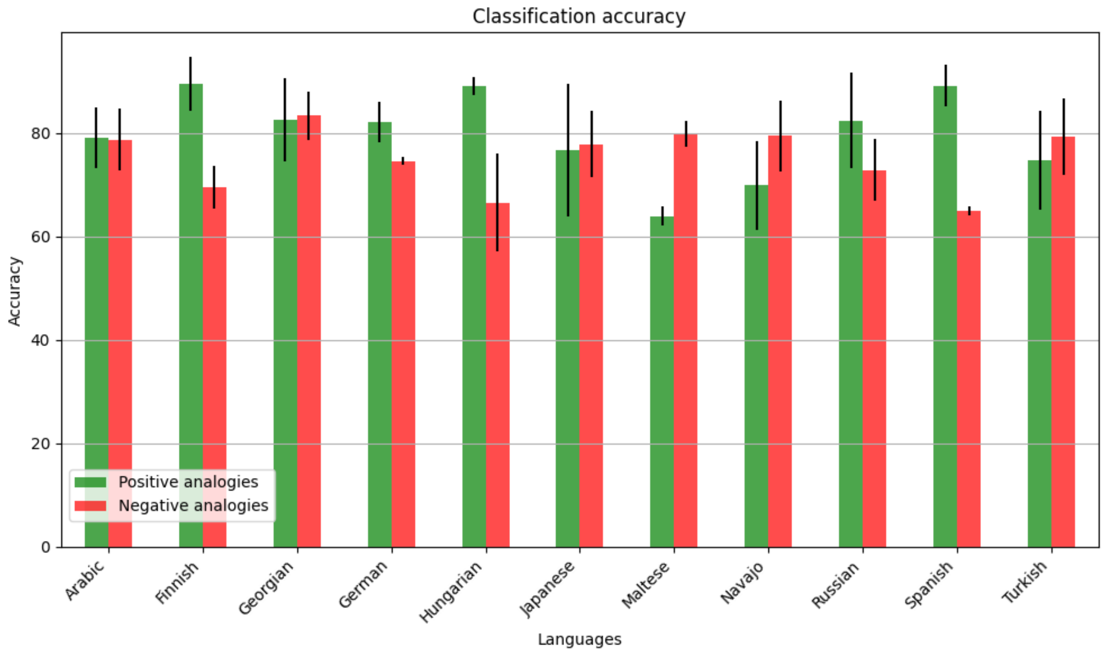


Figure E.5: Accuracy results (in %) for the classification task.

F Previous transfer results

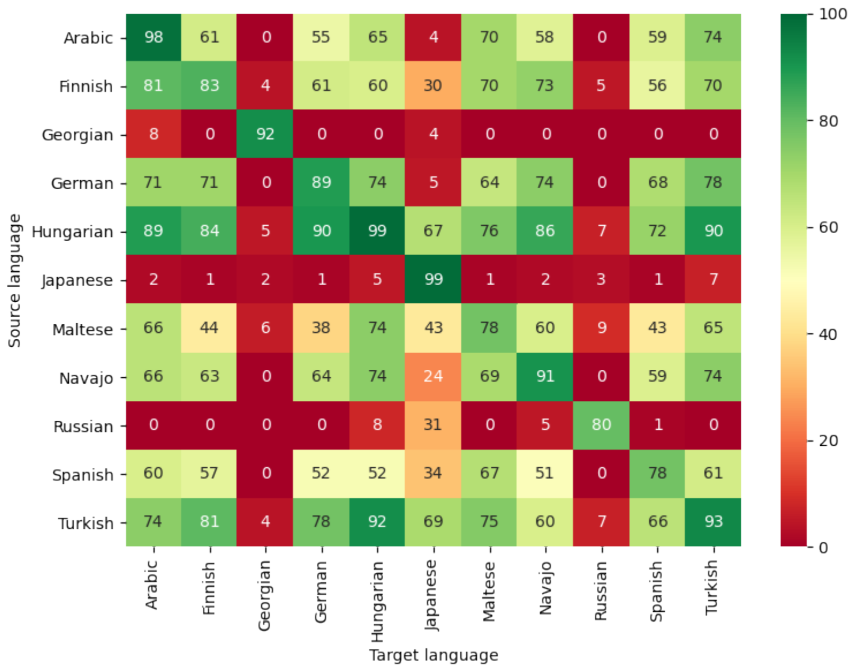


Figure F.6: Transfer accuracy for the negative analogies

G Analogy solving model

As mentioned previously, we use two different analogy solving models. One with shared parameters between the two first fully-connected layers (named *shared parameters*) and one with different parameters (named *different parameters*).

G.1 Different parameters

Table G.3: Accuracy (in %) of 10 runs of the monolingual analogy solving models

	Finnish	German	Hungarian	Spanish	Turkish
Finnish	76.46±1.58	39.75±2.67	80.33±1.95	/	79.89±2.43
German	94.26±0.63	88.70±0.58	70.55±2.61	70.55±2.61	/
Hungarian	43.93±3.44	85.39±1.76	78.72±0.53	85.39±1.76	40.98±3.42
Spanish	/	94.26±0.53	94.26±0.53	91.72±0.43	95.83±0.24
Turkish	64.98±2.76	/	70.74±2.23	94.03±3.70	80.37±1.00

Table G.4: Accuracy (in %) of 5 runs of the bilingual analogy solving models

	Finnish	German	Hungarian	Spanish	Turkish
Finnish	/	66.01±33.01	36.28±0.76	/	28.34±0.48
German	64.43±32.23	/	30.53±0.57	11.92±3.30	/
Hungarian	50.61±2.26	77.98±1.24	/	94.02±0.29	33.89±0.77
Spanish	/	32.42±17.01	78.99±0.13	/	40.45±1.52
Turkish	46.43±1.24	/	16.07±0.90	70.86±0.04	/

Table G.5: Accuracy (in %) of 10 runs of the omnilingual analogy solving model. Bold result indicate a significant improvement with regards to the bilingual models.

	Finnish	German	Hungarian	Spanish	Turkish
Finnish	61.85±1.79	3.22±1.73	28.58±2.69	/	**55.23±3.85**
German	3.15±1.47	64.38±1.27	**66.91±4.49**	**62.07±3.58**	/
Hungarian	36.26±4.52	55.25±1.47	73.33±1.31	78.36±1.33	32.00±3.03
Spanish	/	**61.16±2.54**	74.05±1.77	69.38±1.65	**70.67±4.03**
Turkish	**54.12±1.48**	/	**25.38±3.94**	65.72±6.09	52.23±1.09

G.2 Shared parameters

Table G.6: Shared parameters: Accuracy (in %) of 10 runs of the monolingual analogy solving models

	Finnish	German	Hungarian	Spanish	Turkish
Finnish	76.61±1.15	42.06±2.82	80.13±1.37	/	81.53±1.57
German	94.51±0.60	89.26±0.51	71.94±2.84	71.94±2.84	/
Hungarian	42.99±4.69	84.22±3.33	78.49±0.65	84.22±3.33	40.75±1.89
Spanish	/	93.75±0.38	93.75±0.38	91.18±0.51	96.40±0.42
Turkish	67.43±1.06	/	**73.61±0.77**	95.17±1.12	80.37±0.28

Table G.7: Accuracy (in %) of 5 runs of the bilingual analogy solving models

	Finnish	German	Hungarian	Spanish	Turkish
Finnish	/	83.24±0.35	35.58±0.74	/	28.14±1.14
German	80.13±0.57	/	30.14±0.29	12.92±6.03	/
Hungarian	51.21±3.09	78.09±0.74	/	94.05±0.12	34.55±0.60
Spanish	/	36.79±11.74	79.11±0.63	/	41.83±0.85
Turkish	47.23±0.91	/	15.29±0.85	70.79±0.06	/

Table G.8: Accuracy (in %) of 10 runs of the omnilingual analogy solving model

	Finnish	German	Hungarian	Spanish	Turkish
Finnish	60.95±1.75	3.34±1.16	30.34±2.02	/	53.82±2.74
German	3.05±1.43	63.78±0.88	60.99±4.86	62.65±3.47	/
Hungarian	35.06±4.49	56.35±1.96	71.69±1.84	70.61±7.90	27.10±3.17
Spanish	/	61.90±2.42	67.32±5.05	67.90±1.97	66.12±6.86
Turkish	54.00±2.41	/	24.19±2.21	64.94±5.61	50.39±1.64

G.3 Number of different features per pair of languages

As we can see on Figure G.7, most pairs share very few morphological transformations. We were not able to find a significant link between the accuracy for the omnilingual analogy solving model and the number of different shared features. We can notice however that the results for the pairs (Finnish, Hungarian) and (Hungarian, Finnish) are among the lower ones. These two languages share 19 different transformations, which is the greatest number of shared features between two different languages in our dataset. Extending our model to more languages could allow us to investigate the link between the performance of the model and the number of transformations in the training data.

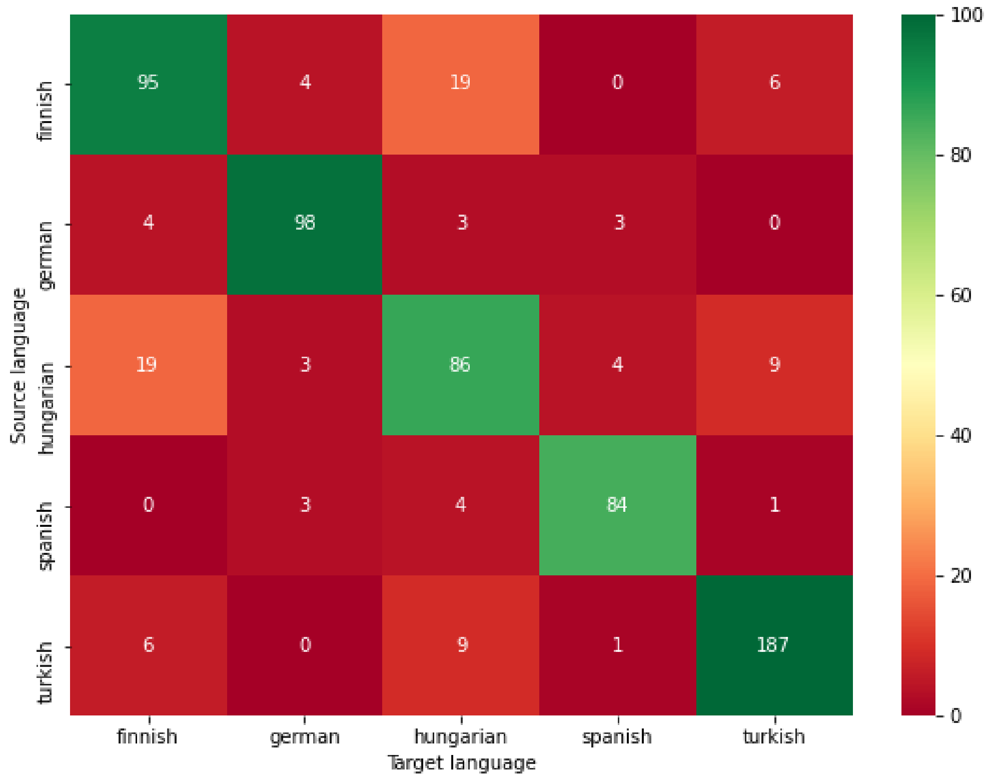
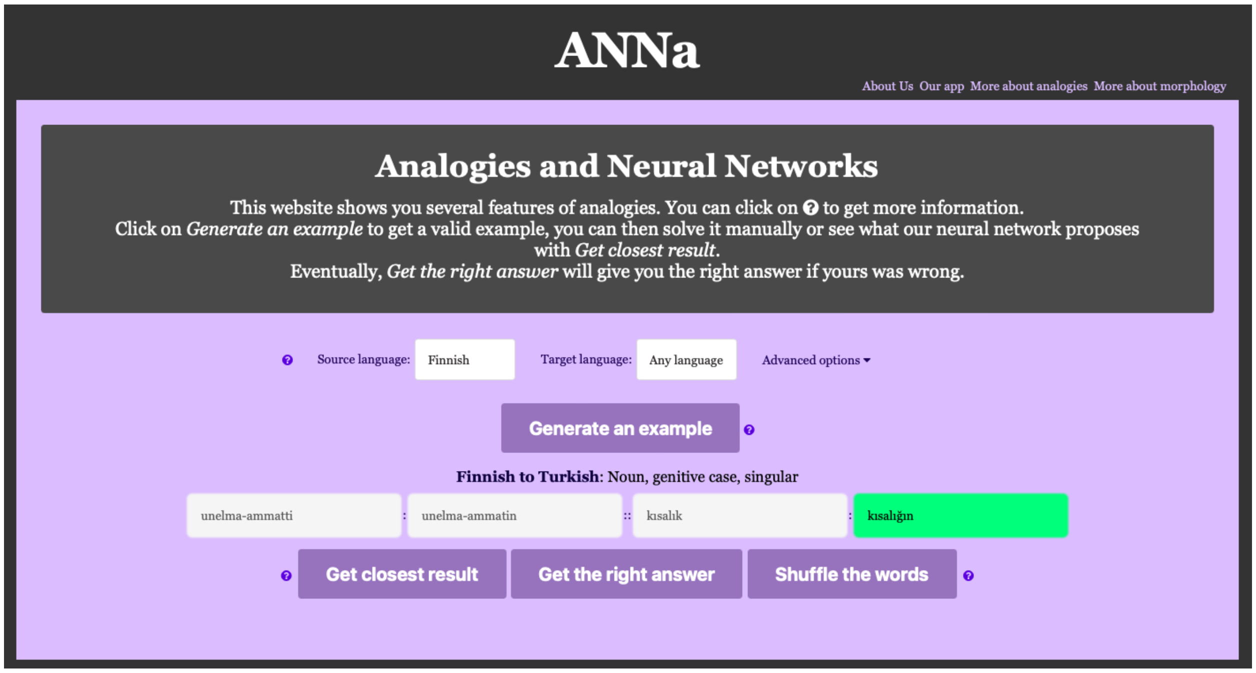


Figure G.7: Number of different features for each pair of languages

H Website preview



Analogies and Neural Networks

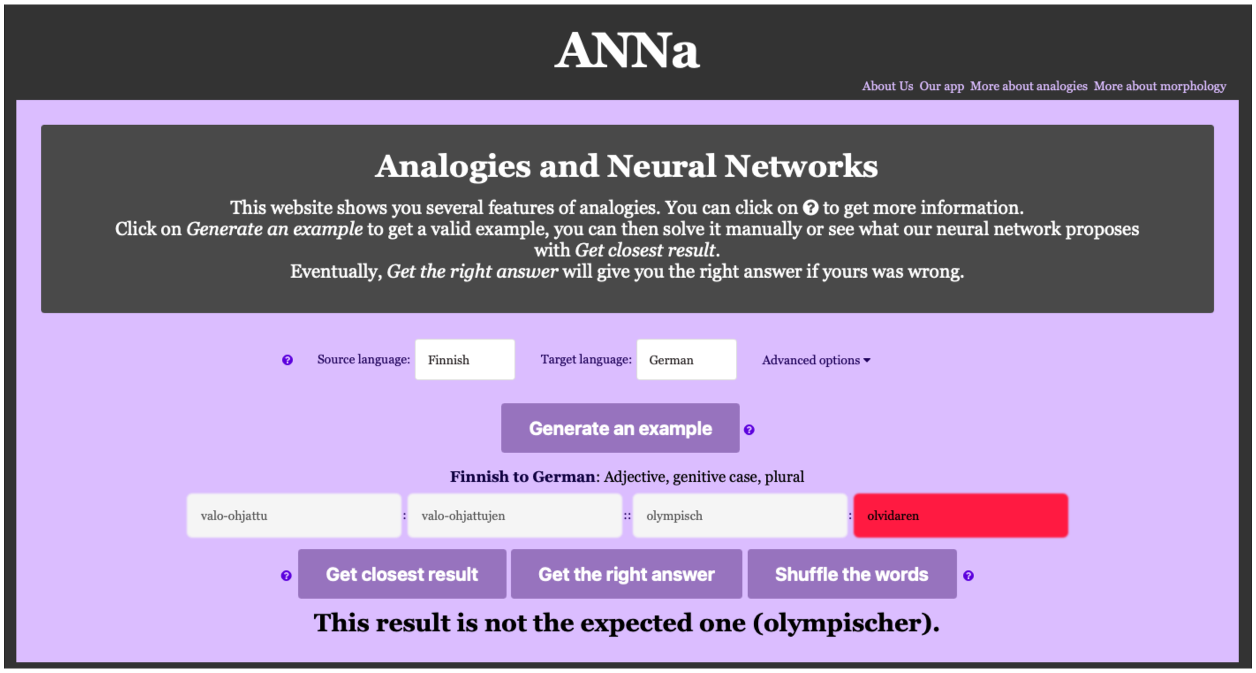
This website shows you several features of analogies. You can click on ? to get more information.
Click on *Generate an example* to get a valid example, you can then solve it manually or see what our neural network proposes with *Get closest result*.
Eventually, *Get the right answer* will give you the right answer if yours was wrong.

Source language: Finnish Target language: Any language

Finnish to Turkish: Noun, genitive case, singular

unelma-ammatti : unelma-ammatin :: kısalık : kısalığın

(a) Correct analogy



Analogies and Neural Networks

This website shows you several features of analogies. You can click on ? to get more information.
Click on *Generate an example* to get a valid example, you can then solve it manually or see what our neural network proposes with *Get closest result*.
Eventually, *Get the right answer* will give you the right answer if yours was wrong.

Source language: Finnish Target language: German

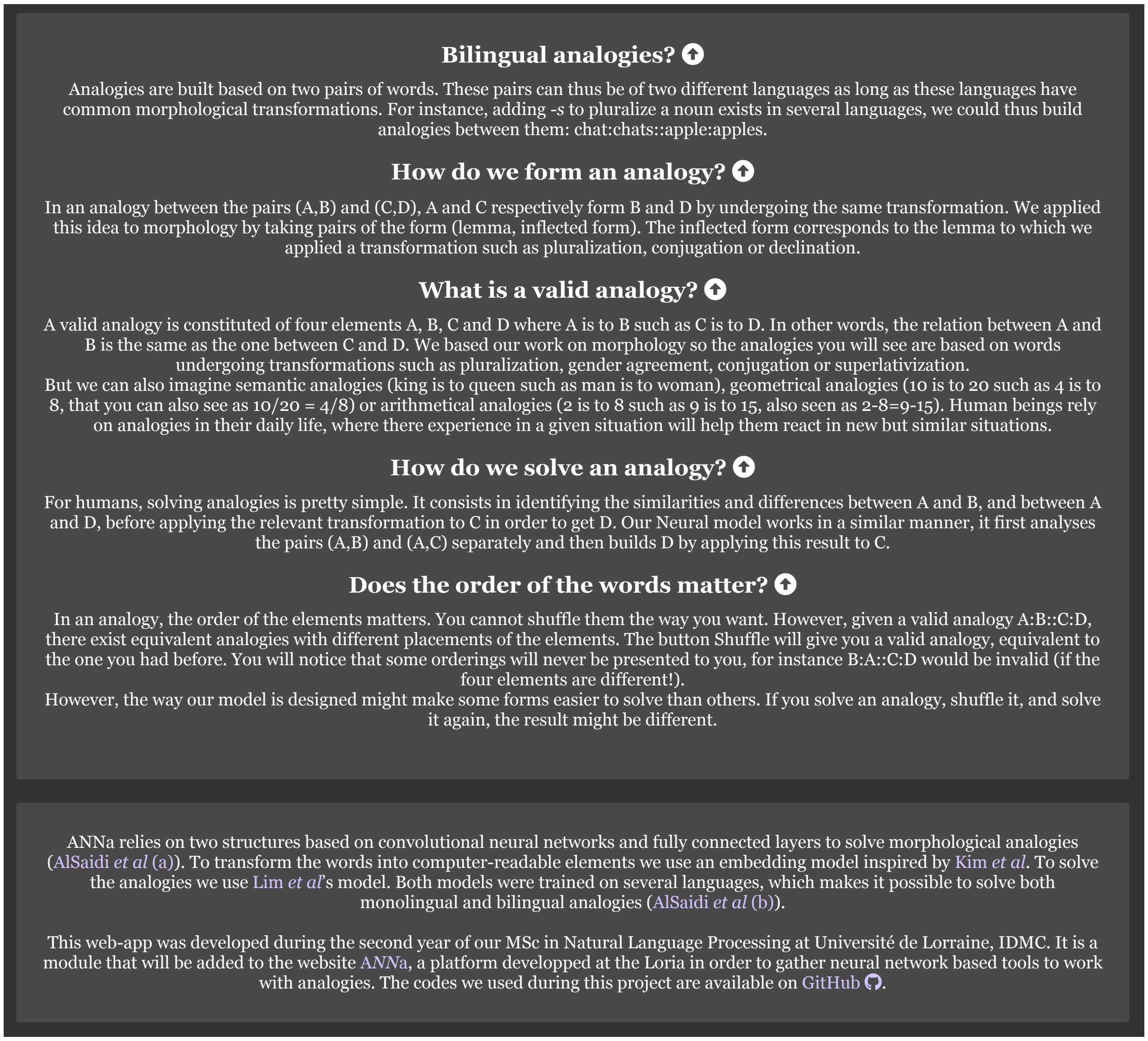
Finnish to German: Adjective, genitive case, plural

valo-ohjattu : valo-ohjattujen :: olympisch : olvidaren

This result is not the expected one (olympischer).

(b) Incorrect analogy

Figure H.8: Preview of the analogy solving section of our software



Bilingual analogies?

Analogies are built based on two pairs of words. These pairs can thus be of two different languages as long as these languages have common morphological transformations. For instance, adding -s to pluralize a noun exists in several languages, we could thus build analogies between them: chat:chats::apple:apples.

How do we form an analogy?

In an analogy between the pairs (A,B) and (C,D), A and C respectively form B and D by undergoing the same transformation. We applied this idea to morphology by taking pairs of the form (lemma, inflected form). The inflected form corresponds to the lemma to which we applied a transformation such as pluralization, conjugation or declination.

What is a valid analogy?

A valid analogy is constituted of four elements A, B, C and D where A is to B such as C is to D. In other words, the relation between A and B is the same as the one between C and D. We based our work on morphology so the analogies you will see are based on words undergoing transformations such as pluralization, gender agreement, conjugation or superlativization.
But we can also imagine semantic analogies (king is to queen such as man is to woman), geometrical analogies (10 is to 20 such as 4 is to 8, that you can also see as 10/20 = 4/8) or arithmetical analogies (2 is to 8 such as 9 is to 15, also seen as 2-8=9-15). Human beings rely on analogies in their daily life, where there experience in a given situation will help them react in new but similar situations.

How do we solve an analogy?

For humans, solving analogies is pretty simple. It consists in identifying the similarities and differences between A and B, and between A and D, before applying the relevant transformation to C in order to get D. Our Neural model works in a similar manner, it first analyses the pairs (A,B) and (A,C) separately and then builds D by applying this result to C.

Does the order of the words matter?

In an analogy, the order of the elements matters. You cannot shuffle them the way you want. However, given a valid analogy A:B::C:D, there exist equivalent analogies with different placements of the elements. The button Shuffle will give you a valid analogy, equivalent to the one you had before. You will notice that some orderings will never be presented to you, for instance B:A::C:D would be invalid (if the four elements are different!).
However, the way our model is designed might make some forms easier to solve than others. If you solve an analogy, shuffle it, and solve it again, the result might be different.

ANNa relies on two structures based on convolutional neural networks and fully connected layers to solve morphological analogies (AlSaidi *et al* (a)). To transform the words into computer-readable elements we use an embedding model inspired by Kim *et al*. To solve the analogies we use Lim *et al*'s model. Both models were trained on several languages, which makes it possible to solve both monolingual and bilingual analogies (AlSaidi *et al* (b)).

This web-app was developed during the second year of our MSc in Natural Language Processing at Université de Lorraine, IDMC. It is a module that will be added to the website ANNa, a platform developped at the Loria in order to gather neural network based tools to work with analogies. The codes we used during this project are available on GitHub.

Figure H.9: Preview of the explanation section of our software