MSc Natural Language Processing –2020 - 2021
UE 705 –Supervised Project

A Neural Approach to Detecting and Solving Morphological Analogies across Languages

Students:
Safa Alsaidi
Amandine Decker
Puthineath Lay

Supervisors:
Miguel Couceiro
Esteban Marquer
Reviewer:
Maxime Amblard

June 17, 2021

Contents

Acknowledgements

The achievement of this project could not have been possible without the contributions from our respective people. We are highly thankful to our supervisors, Miguel Couceiro and Esteban Marquer for their practical guidance, constructive advice and endless support throughout this year. We would also like to thank Timothee Mickus for providing us with the Japanese Dataset, which we used extensively in our empirical setting. Last but not least, we thank Pierre-Alexandre Murena for his guidance, willigness to share his knowledge, and eagerness to help us. Lastly, we are extremely grateful to all lecturers of IDMC for teaching us to gain a deeper understanding on developing this project.

Introduction

Analogy consists 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. Analogy can be used as a method of reasoning and can be expressed by an analogical proportion, which is a statement such as "A is to B as C is to D". An analogical proportion becomes an equation if one of its four objects is unknown (Miclet et al., 2008).

Analogies have been extensively studied in Natural Language Processing, which resulted in different formalizations with noteworthy applications in various domains such as derivational morphology (Murena *et al.*, 2020). Analogies on words can refer exclusively to their morphology as in the following example:

$$R = \{x \,|\, \text{apple is to tree as apples is to } x\}.$$

Solving this form of equations can be done by calculating the set R of solutions x which satisfy the analogy (Miclet et al., 2008). In this case, the observed solution is based on morphological variations of two words: "apple" and "tree". 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 (Murena *et al.*, 2020; Lim et al., 2019; Lepage, 2003). Recent empirical works propose a data-oriented strategies to learn the correctness of analogies from past observations. These strategies are based on machine learning approaches.

In this project we focused on analogies, with a particular emphasis on morphological word variations and how they determine the relation between different words. Our project was inspired by the paper of Lim et al. (2019), which proposes a deep learning approach to train models of semantic analogies on word embeddings [1] by using pre-trained GloVe [2](Pennington et al., 2014) embeddings. The results of this approach were competitive for analogy classification and solving. This led us to adapting the neural framework presented in Lim et al. (2019) to morphological analogies. The major difference is that our morphological approach relies on customized model embeddings, which were fully designed and trained during this project.

We used the SIGMORPHON 2016 dataset (Cotterell *et al.*, 2016) as well as the JAPANESE BIGGER ANALOGY TEST SET dataset released by Karpinska *et al.* (2018) to analyze how deep learning could help us classify and solve morphological analogies. These datasets covered 11 languages in total, 10 of which are from the SIGMORPHON 2016 dataset (Cotterell *et al.*, 2016), while another one is from JAPANESE BIGGER ANALOGY TEST SET (Karpinska *et al.*, 2018). We used a character-level embedding to encode the morphemes of the words. To emphasize, we focus on the structure of the words more than their meanings. Once the model is trained, it is be able to embed any word, even those that did not appear during the training phase.

[1] A word embedding is word representation that allows words with similar meaning to have a similar representation in vector space

[2] GloVe is an open-source project at Stanford used for obtaining word embeddings

This report is organized as follows. We start by introducing analogical proportions, motivation, objective, and related work. We also introduce the datasets we used and discuss their properties in Chapter 1. Chapter 2 introduces the approaches that we used to classify and solve morphological analogies. We also discus some results we obtained and some of the challenges we encountered. Chapter 3 illustrates the adaptability of our framework by showing that models trained on a given language may be *transferred* to a different language. In fact, the latter revealed noteworthy (dis)similarities between languages. In Chapter 4, we briefly discuss the results that we have obtained and describe the current status of the project. Apart from that, we also highlight some of the challenges encountered through the development process and present some topics of future work, namely, potential improvements to the current project.

To realize this project, we used exclusively Python (version 3.9) (Van Rossum & Drake, 2009) and in particular the deep learning-oriented library PyTorch (Paszke et al., 2019). We used various built-in Python libraries including Pandas, Matplotlib, Numpy and Scikit-learn for minor uses like storing data and support functions. To train the deep leaning models, we required substantial computational resources, so experiments presented in this paper were carried out using the Grid'5000 (Balouek *et al.*, 2013) testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see `https://www.grid5000.fr`). All the codes we used during this project are available on GitHub (see `https://github.com/AmandineDecker/nn-morpho-analogy.git`).

Chapter 1

Definition of the problem

Analogical learning based on formal analogy can be applied to many problems in computational linguistics. To quote Haspelmath (2002): "Morphology is the study of systematic co-variation in the form and meaning of words." When analyzing analogy in a morphological approach, we are looking at the co-variation in the form of a single word. For example, "reader is to doer as reading is to doing" is an analogy made of four tuples that present the different variations of the lexicons "read" and "do" (Miclet & Delhay, 2003). Analyzing analogies based on morphology allows the linguist to find how the word could vary based on gender, plurality, tense, mood, *etc.* It also allows the linguist to predict how words change form based on these classified patterns even if they are not familiar with certain words. When it comes to real life application, morphological analogies are used as a method for acquiring new languages. It was one of the evaluating methods used in assessment tests like SAT, TOEFL, and ACT, making around 20 percent of the questions posed (Turney, 2001; Betrand, 2016). These questions mainly focus on inflectional affixes in morphology. Therefore, in this project we aimed to:

- build a model that automatically determines if four words form a valid analogy;
- build a model which can solve morphological analogical equations;
- determine whether different languages share morphological properties.

As mentioned, for this project we adapted a novel approach, particularly a deep learning one, to deal with morphological analogies. Various models have been proposed to solve semantic analogies including Textual Analogy Parsing approach (TAP), Dependency Relation approach (DP), Vector Space Model (VSM), and Neural Network approach (NN) (Lamm *et al.*, 2018; Chiu et al., 2007; Turney, 2006; Lim et al., 2019). Out of these models, we were most interested in VSM and NN approaches, and we finally decided to work with a NN and develop Lim's approach (Lim et al., 2019) due to the interesting results that they achieved. Though this approach is more complex than VSM, it is in fact not that complex to implement to solving morphological analogies. We adapted Lim *et al.*'s approach by customising morphological word embeddings as, to our knowledge, there were no ready made ones for this task. Thus we trained and developed an embedding model for this project.

1.1 Tasks to address

The two tasks tackled by (Lim et al., 2019) are identification of analogical proportions and solving of analogical equations, which are both based on natural language. Words are encoded

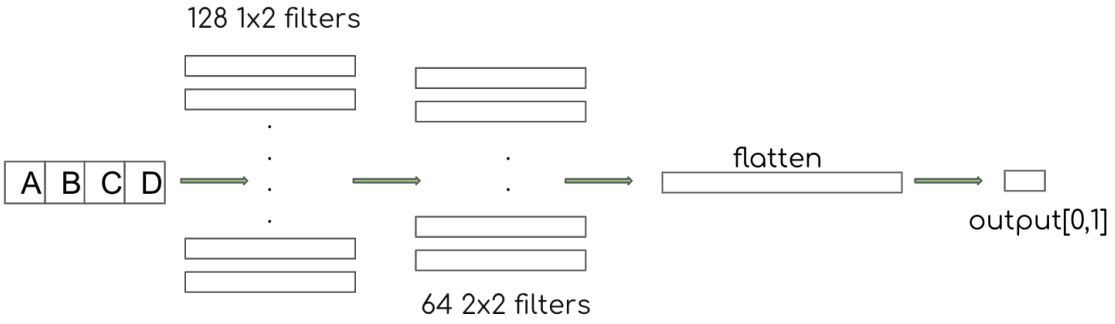


Figure 1.1: Structure of the CNN as a classifier. (Lim et al., 2019)

as numerical vectors with a word embedding process.

In terms of neural networks, identifying an analogical proportion consists in deciding whether $A : B :: C : D$ holds true or not: it is a binary classification task. Solving analogical proportions can be seen as a regression task where we try to approximate the function f such that $f(A, B, C) = D$ if $A : B :: C : D$ holds true.

Both tasks are further explained in the following subsections. In our project, we worked on both of these tasks, where we train our custom embedding model on the classification task and then use it to solve morphological analogies.

1.1.1 Classification

The classification task consists in determining if a quadruple (A, B, C, D) is a valid analogy or not. The model is based on the structure provided in (Lim et al., 2019), the network can be visualised on FIGURE 1.1.

As input of the model we have four vectors A, B, C, and D of size n. We stack them to get a matrix of size $n \times 4$. This matrix is the representation of the analogy $A : B :: C : D$, it means that it should be analysed as a structure containing two pairs of vectors: (A, B) and (C, D).

The size $h \times w : 1 \times 2$ of the filters of the first CNN layer respects the boundaries between the two pairs. This layer should analyse each pair and be able to notice the differences and similarities between the elements of each pair.

The second CNN layer should compare the results of the analysis of both pairs: if A and B are different in the same way as C and D then $A : B :: C : D$ is a valid analogy.

Eventually all the results are flattened and used as input of a dense layer. We use a sigmoid activation to get a result between 0 and 1 as we work with a binary classifier.

1.1.2 Solving analogies

Given the words A, B and C, solving analogical equations consists in producing the word D such that $A : B :: C : D$ holds true. From a machine learning perspective, a model which can perform this task would take as input a triple $(embed(A), embed(B), embed(C))$[1] and would produce as an output a X, which ideally would correspond to the embedding of the word D

[1] $embed(X)$ refers to the embedding of the word X

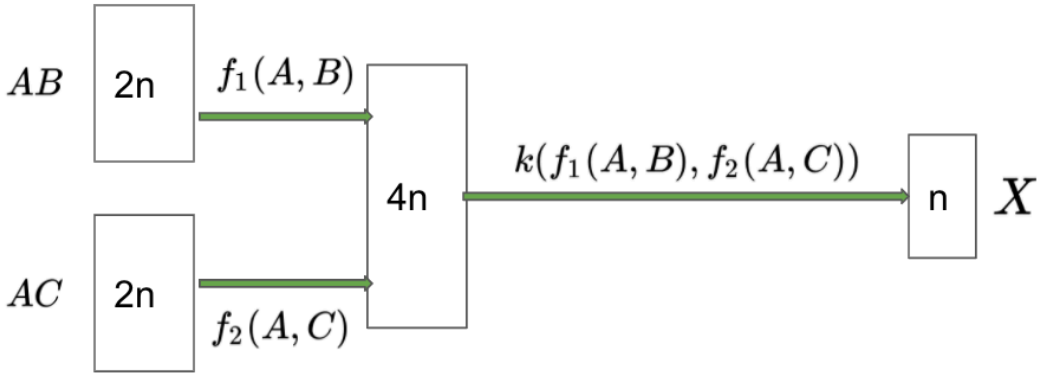


Figure 1.2: Structure of the neural network for analogy solving. (Lim et al., 2019)

such that $A : B :: C : D$ holds true. The analogy solving task thus requires two models: one producing a vector based on three others, like what we just described, and also an embedding model which we discuss in Sections 2.1 and 2.2.

The model we used to produce D given (A, B, C) is taken from (Lim et al., 2019). The assumption of the authors is that the X of the analogical equation $A : B :: C : X$ can be found thanks to a function f of A, B and C. This point of view redefines the analogy solving task. It becomes a multi-variable regression problem based on the following dataset:

$$\left\{ ((embed(A), embed(B), embed(C)), embed(D)) \;\middle|\; \begin{array}{l} A : B :: C : D \text{ belongs to the original} \\ \text{set of valid analogies} \end{array} \right\}$$

The neural network they designed is described on Figure 1.2. The structure reflects the relevance of the relations between the different words of the analogy. Indeed, for an analogy $A : B :: C : X$, the link between A and B as well as the link between A and C is relevant to determine X but also the relation between the pairs (A, B) and (A, C). This is the reason why the function $g(embed(A), embed(B), embed(C))$, which determines X, is approximated by two hidden functions $f_1(embed(A), embed(B))$ and $f_2(embed(A), embed(C))$. Overall, we have:

$$X = g(f_1(embed(A), embed(B)), f_2(embed(A), embed(C))).$$

In the neural network, f_1 and f_2 are approximated by two linear layers. The input is of size $2 \times n$, where n is the embedding size, because the inputs are the concatenation of two embeddings

$$embed(A) \text{ and } embed(B) \quad \text{or} \quad embed(A) \text{ and } embed(C).$$

The output of each of these layers, of size $2 \times n$, are concatenated into a matrix of size $4 \times n$ and fed to a final linear layer which approximates g. The output is of size n which is the size of an embedding so we can expect the model to produce a vector corresponding to the embedding of X such that $A : B :: C : X$.

1.2 Properties of analogies for data augmentation

Deep learning approaches require a large amount of data. Therefore we took advantage of some properties of analogies to produce more data based on our datasets, this process is called *data augmentation*. Given one valid analogy, we can generate seven more valid analogies and three invalid ones. Training our models on different equivalent forms of the same analogy helps reducing overfitting. In this section, we describe the properties which enable us to augment our datasets.

Analogies are classified and grouped based on the type of relation that exist between word pairs. The first implementation of proportions was introduced by Ancient Greeks and was used in the domain of numbers. Two examples worth mentioning are arithmetic proportion and geometric proportion (Couceiro *et al.*, 2017). These two examples illustrate the analogical proportion statement of "A is to B as C is to D."

- A, B, C, and D are proportional if $A - B = C - D$ (arthimetic);
- A, B, C, and D are proportional if $\frac{A}{B} = \frac{C}{D}$ (geometric).

These quaternary relations obey the following axioms (Lepage, 2003):

(1) $A : B :: A : B$ (reflexivity);

(2) $A : B :: C : D \rightarrow C : D :: A : B$ (symmetry);

(3) $A : B :: C : D \rightarrow A : C :: B : D$ (central permutation);

From these properties, we can infer the following 8 equivalent analogies to $A : B :: C : D$ (Cortes & Vapnik, 1995; Gladkova et al., 2016; Delhay & Miclet, 2004).

- $A : B :: C : D$ (base form).
- $C : D :: A : B$ (symmetry).
- $A : C :: B : D$ (central permutation).
- $B : A :: D : C$.
  Proof. $A : B :: C : D \overset{(3)}{\Longrightarrow} A : C :: B : D \overset{(2)}{\Longrightarrow} B : D :: A : C \overset{(3)}{\Longrightarrow} B : A :: D : C$;
- $D : B :: C : A$.
  Proof. $C : D :: A : B \overset{(3)}{\Longrightarrow} C : A :: D : B \overset{(2)}{\Longrightarrow} D : B :: C : A$.
- $D : C :: B : A$.
  Proof. $B : A :: D : C \overset{(2)}{\Longrightarrow} D : C :: B : A$;
- $C : A :: D : B$.
  Proof. $D : B :: C : A \overset{(2)}{\Longrightarrow} C : A :: D : B$;
- $B : D :: A : C$.
  Proof. $A : C :: B : D \overset{(2)}{\Longrightarrow} B : D :: A : C$.

Fremdsprache	pos=N, case=ACC, gen=FEM, num=PL	Fremdsprachen

Figure 1.3: Example from the German training set for task 1.

In addition to these forms, we have 3 analogical forms which are considered invalid analogies as they cannot be deduced from the base form $A : B :: C : D$ and the axioms **(1)**, **(2)**, and **(3)** and they contradict the intuition:

1. $B : A :: C : D$;
2. $C : B :: A : D$;
3. $A : A :: C : D$.

We used these properties to augment our datasets for the classification and solving tasks: we produced eight valid analogies and three invalid ones given a valid $A : B :: C : D$ for the classification task and only the eight valid ones for solving analogies.

1.3 Our dataset(s)

For this project, we used the SIGMORPHON 2016 dataset (Cotterell *et al.*, 2016) and the JAPANESE BIGGER ANALOGY TEST SET (Karpinska *et al.*, 2018). As previously mentioned, these datasets covered 11 languages in total, 10 of which are from the SIGMORPHON 2016 dataset while the other one is from JAPANESE BIGGER ANALOGY TEST SET. In this chapter, we introduce both of these datasets and explain some of the preprocessing conducted before the realization part of the project.

1.3.1 SIGMORPHON 2016 dataset

One of the datasets we used was SIGMORPHON 2016 (Cotterell *et al.*, 2016), which contained training, development and test data. Data is available for 10 languages: Spanish, German, Finnish, Russian, Turkish, Georgian, Navajo, Arabic, Hungarian and Maltese. Most of them are considered as languages with rich inflection (Cotterell *et al.*, 2018). It is separated in 3 subtasks: inflection, reinflection and unlabled reinflection. All the provided files are in UTF-8 encoded text format. Each line of a file is an example for the task, the fields are separated by a tabulation. In our experiments, we focused 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 morphosyntactic descriptions (MSD), *e.g.,* the grammatical properties of the words such as their part of speech, case or number (among others).

A triple LEMMA, MSD, TARGET FORM from the German training data is presented on Figure 1.3

1.3.2 Japanese Bigger Analogy Test Set

The dataset we used for Japanese was generated from pairs of words of the dataset released by Karpinska *et al.* (2018). This dataset contains several files, each of them containing pairs of linguistically related words. The list of relations is described in Table 1.1.

	Relation	Example	Pairs
Inflectional morphology	verb_dict - mizenkei01	会う → 会わ/あわ	50
	verb_dict - mizenkei02	出る → 出よ/でよ	51
	verb_dict - kateikei	会う → 会え/あえ	57
	verb_dict - teta	会う → 会っ/あっ	50
	verb_mizenkei01 - mizenkei02	会わ → 会お/あお	50
	verb_mizenkei02 - kateikei	会お → 会え/あえ	57
	verb_kateikei - teta	会え → 会っ/あっ	50
	adj_dict - renyokei	良い → 良く/よく	50
	adj_dict - teta	良い → 良かっ/よかっ	50
	adj_renyokei - teta	良く → 良かっ/よかっ	50
Derivational morphology	noun_na_adj + ka	強 → 強化/きょうか	50
	adj + sa	良い → 良さ/よさ	50
	noun + sha	筆 → 筆者/ひっしゃ	50
	noun + kai	茶 → 茶会/ちゃかい	50
	noun_na_adj + kan	同 → 同感/どうかん	50
	noun_na_adj + sei	毒 → 毒性/どくせい	52
	noun_na_adj + ryoku	馬 → 馬力/ばりき	50
	fu + noun_reg	利 → 不利/ふり	50
	dai + noun_na_adj	事 → 大事/だいじ	50
	jidoshi - tadoshi	出る → 出す/だす	50

Table 1.1: List of relations between the words of the Japanese dataset.

We were interested in inflectional and derivational morphology relations for which 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. This set is smaller than the SIGMORPHON 2016 one, but this was not an issue when training the classification and embedding model because we could produce 8 valid and 3 invalid analogies in total with a given valid one. However, Japanese produced poor results when solving analogies which may be related to the size of the dataset.

1.3.3 Loading and augmenting the data

To obtain morphological analogies from the SIGMORPHON 2016, we defined our analogical proportions as follows: for any two triples of the form:

$$\langle A, F, B\rangle, \langle A', F', B'\rangle$$

which share the same morphological features ($F = F'$), we considered $A : B :: A' : B'$ an analogical proportion. Figure 1.4 presents two examples from the German training set for building analogies.

Notice that only one analogy is generated for each pair of triples. For example, if we generate $A : B :: A' : B'$, we do not generate $A' : B' :: A : B$ as it is generated by the process introduced in Section 1.2. During training and evaluation, for each sample of the dataset we generated 8 positive and 3 negative examples following the properties mentioned in Section 1.2. This approach does not match the one of Lim *et al.*, as they additionally generated the 8 equivalent forms for each negative example.

The SIGMORPHON 2016 dataset contains training and testing files for all of the languages.

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

"Fremdsprache":"Fremdsprachen"::"Absorption":"Absorptionen" is a valid analogy.
"Fremdsprache":"Fremdsprachen"::"absurd":"absurder" is not because ("Fremdsprache", "Fremdsprachen") and ("absurd", "absurder") do not share the same MSD.

Figure 1.4: Examples from the German training set for building analogies.

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

Table 1.2: Number of analogies for each language before data augmentation.

For our project, we generated analogies by using these files. The number of analogies for each language is presented in Table 1.2. For both the training and evaluation, we decided to work with 50,000 analogies to keep the training time reasonable (around 6 hours on Grid'5000). Maltese, Navajo and Turkish were evaluated on less than 50,000 analogies because the related datasets were too small as we can see in Table 1.2.

The Japanese dataset contains one file per transformation (listed in Table 1.1). We grouped all these files together in one file following the same format of the SIGMORPHON 2016 files: LEMMA, TRANSFORMATION, TARGET FORM. When we load the data, analogies are loaded based on word pairs with the same relation, which yields 26410 analogies. We split the dataset to get 70 percent for training and 30 percent for testing. For reproducibility, the list of analogies for training and testing were stored in separate files to ensure that the same sets were used every time we evaluated.

To load the data and build the mentioned analogies, we used the "data.py" code provided by Esteban Marquer. The "data.py" file contains the classes we use to import the data and transform the words into vectors. The code can be manipulated using different modes ("train", "dev", "test", "test-covered") and different languages ("Arabic", "Finnish", "Georgian", "German", "Hungarian", "Japanese", "Maltese", "Navajo", "Russian", "Spanish", "Turkish"). An augmentation function is introduced which, given an analogy, yields all the equivalent forms based on the properties described in Section 1.2. Another function generated the invalid forms described in Section 1.2. When we instantiate "Task1Dataset" class from the "data.py" file, it generates a list of quadruples where each quadruple represents an analogy. The quadruples can be in either plain words or lists of integers. We used plain words with the GloVe pre-trained embeddings, *e.g.,* for German only (Section 2.1). The lists of integers were used with our custom embedding model. These lists are built with a dictionary mapping characters to integers: we

first build the list of all the characters contained in the file and assign for each of them an integer (*e.g.,* an ID), then the encoding of a word is the list of IDs corresponding to the characters of the word. For instance the German word "abfällig" is encoded as "[21, 22, 26, 50, 32, 32, 29, 27]".

The dictionary thus depends on the input file. Note that the size and the content of the embedding layer of our model depends on the size and the content of the dictionary, *i.e.,* we cannot use a file with a dictionary of size m with an embedding model of size n if $n \neq m$. Moreover, if the dictionary has the right size but the IDs are not matched with the same letters as in the dictionary used during training, the results can be unexpected. This topic is discussed later in Section 2.2.

1.3.4 (Dis)similarities between languages

In order to compare the different languages, and have more material to explain our results later, we computed some statistics on the datasets.

1.3.4.1 Statistics about the words of the languages

Words length For all the languages except Japanese, the mean of the words (Appendix B and Table B.1) length lies between 7.6 and 11. For Japanese, the words mean length is of 5 ± 3 (mean length $\pm$ standard deviation), this dataset contains very short words (one or two characters) as well as longer ones (up to eighteen characters).

Differences between training and testing set For all the languages except Japanese, between 58% and 87% of the words of the test set are new compared to the words of the training set (Table B.2). It means that the models were evaluated on new analogies but also partially on analogies based on words never encountered. It was not the case for Japanese because of the way we built the dataset. Indeed, we first generated all the possible analogies based on the pairs of words we had and then split them between a training and a test set. If we had split the pairs of words instead, we would have had new words in the test set but the training and test sets would have been smaller than they currently are. As a comparison, the datasets from SIGMORPHON 2016 contain at least 104,883 analogies for training against 18,487 for the JAPANESE BIGGER ANALOGY TEST SET.

Number of transformations We call a *transformation* the process used to go from the first word to the second in a word pair. For languages of SIGMORPHON 2016 they are described by an MSD and for Japanese they are the relations described in Table 1.1. Most languages have less than 100 different transformations in the training set (Table B.1) and between 100 and 200 word pairs per transformation in average (Figure B.2). Arabic and Turkish have more transformations (187 and 223) and fewer pairs per transformation (54 and 65 in average) but since there are $\frac{n(n+1)}{2}$ analogies generated for n word pairs with a given transformation, it should not impact the learning process. For Maltese however, there are more than 3,000 different transformations for an average of 5 word pairs per transformation. If the morphemes implied in these transformations are very different for each other it could be a problem for the model as there is very few data for each transformation but a lot of different patterns to learn. Moreover, the test set of Maltese contains 105 new transformations compared to the test set which could be a problem if these transformations imply different modifications from the ones involved in the transformations of

the training set as the model would never have encountered them. Arabic, German and Russian test set also contain new transformations compared to their respective testing sets but to a lesser extent (6 for Arabic, 2 for German and Russian).

Levenshtein distance The Levenshtein distance (Levenshtein, 1965) between two strings is the minimal number of characters to edit (add, delete or modify) to change one string into the other. We computed this distance on each pair of words of the datasets (Figure B.3) in order to evaluate the amount of differences between the first and second word. Our assumption was that a larger distance can imply more complex transformations and thus require a more complex model. The average distance goes from 1.7 to 7.0. The languages with the smallest distance are Georgian, German, Russian and Spanish and the ones with the biggest distance are Japanese and Maltese. The range of Japanese words length is rather wide which could explain a high Levenshtein distance if one word of the pair is short and the other long.

1.3.4.2 The sets of character dictionaries

The first step when we embed a word with our model consists in encoding this word with a list of IDs. Each ID corresponds to a character present in the dataset, the German word "abfällig" is encoded as "[21, 22, 26, 50, 32, 32, 29, 27]. We call character dictionary the mapping from the characters to the IDs for one dataset. The character dictionaries vary from one dataset to another as they contain only the characters used in the dataset.

Character dictionary lengths If we exclude Japanese which uses 632 characters, the lengths of the set of characters in each dataset vary from 30 to 58 (Table B.2). In German, nouns start with a capital letter. German thus uses the biggest character dictionary as it is the only one containing capital letters. The Arabic is also longer than most because of the accented characters.

Comparisons between the character dictionaries of the languages For Finnish, Maltese and Russian, the test set contains one character absent from the training set. To get the full view of the differences in dictionaries between the languages, we computed the coverage of the test dictionaries by the training dictionaries (Figure B.4). We computed it this way because the models are trained on the training sets (and thus learn the related characters) and then evaluated on the test sets (and thus need to deal with the characters of the test set). For a given language, training to test coverage is 100% except for Finnish, Maltese and Russian for which it is 97%. In average, all the languages except Georgian, Japanese and Russian have a coverage of more than 45% (if we do not take Georgian, Japanese and Russian into consideration it rises up to 60%) on other languages. Georgian, Japanese and Russian use a different alphabet than the other languages which explains their poor coverage *of* and *by* other languages.

1.3.4.3 Language families

We investigated the proximity of the languages we work with thanks to their families. (Cole & Siebert-Cole, 2020) provides the family tree of 10 out of 11 languages. Maltese is not represented but based on (Harwood, 2021) we chose to represent it next to Arabic in the Afro-Asiatic family as a Semitic language. In Figure 1.5, the languages are linked to their group or family which are also linked together. Languages closer together belong to the same family as for Spanish and German. Groups and families are also linked to one another depending on their history.

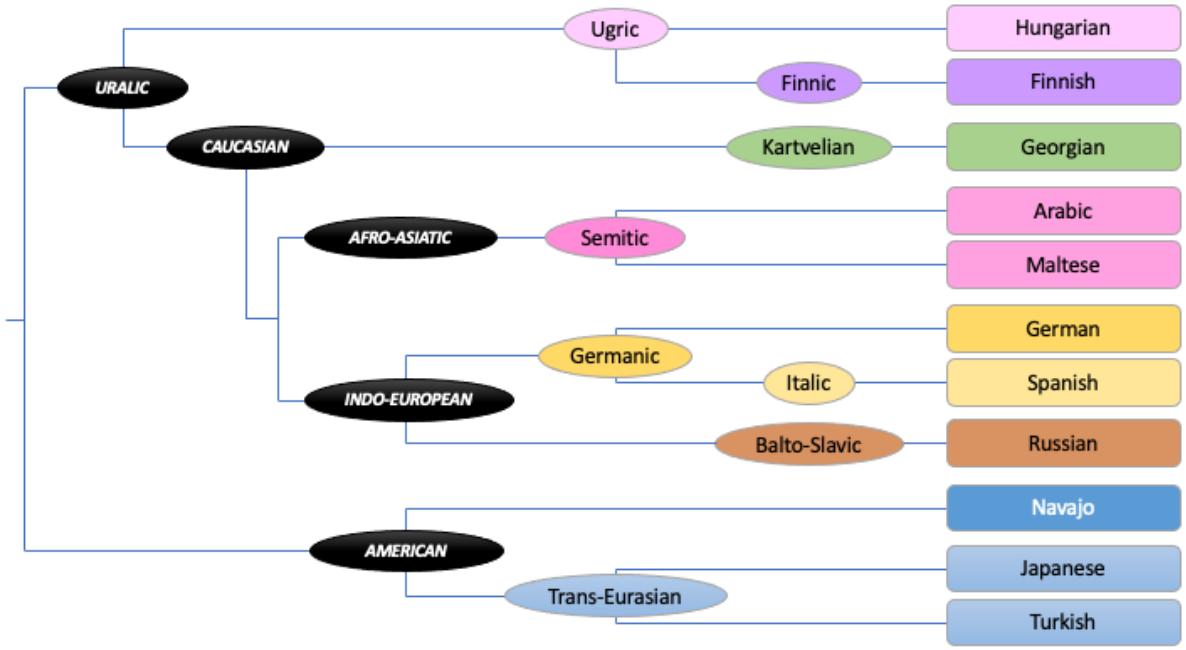


Figure 1.5: Language tree of the 11 languages in the datasets.

For example, Georgian belongs to the Caucasian family, which descends from the Uralic family that includes Hungarian and Finnish. Languages from the Afro-Asiatic family are closer to the languages of the Uralic family than those of the American family.

Chapter 2

Solving analogy related tasks

Before introducing this chapter, we are proud to say that most of the results discussed in Sections 2.2 and 2.2.2 led to the writing of a paper which was submitted to the IEEE International Conference on Data Science and Advanced Analytics 2021.

For our project, we based our experiments on the two tasks described in Lim et al. (2019), *i.e.,* analogy classification and solving of analogical equations, which requires a way to represent words numerically. They worked with the pre-trained word embeddings for English provided by GloVe Pennington et al. (2014). However, we worked with 11 languages (10 from SIGMORPHON 2016 as well as Japanese (Karpinska *et al.*, 2018)) for which GloVe embeddings are not necessarily available. We could have used similar tools which cover more languages such as Grave *et al.* (2018) which covers 157 languages. However, our experiments with GloVe (Section 2.1) showed us that pre-trained word embeddings model tend not to cover the entirety of our dataset. Moreover, these classical word embeddings are trained on word co-occurrence among the training texts, they are thus usually able to solve semantic analogies such as "man'' : "$woman''$ :: "$king''$: X through co-occurrence similarities. But we dealt with morphological analogies and, for most languages, morphology is not sufficiently linked to semantics for these models to perform as good as we would like.

For all these reasons we developed a custom word embedding model focused on morphology. Our model was trained along the classifier for a given language. The idea was to learn sub-words and morphemes from single words (as opposed to texts for classical word embedding models). This difference enabled us to embed any word even if it was not encountered during the training phase, and to model morphology through the sub-words and morphemes.

2.1 First attempt with pre-trained GloVe embeddings

Our first experiment consisted in using GloVe embeddings with the regression model for which Esteban Marquer wrote the code following the structure of Lim et al. (2019). We used the German data of the first task of SIGMORPHON 2016 to build a set of valid analogies with the method described in Section 1.3.3. We obtained a set of $994,740$ quadruples of the form $\langle$"abchasisch", "abchasischerem", "abchasisch", "abchasischerem"$\rangle$ which correspond to valid analogies: a quadruple $\langle A, B, C, D\rangle$ corresponds to the valid analogy $A : B :: C : D$.

When the data is loaded, each word of the quadruple is embedded with GloVe into vectors of size $n = 300$. We chose $n = 300$ as it is the biggest embedding size used by Lim et al. (2019) and a bigger size most likely provides better results.

As explained in Section 1.2, given one valid analogy, we are able to generate 7 more with permutations. Our dataset thus contains $8 \times 99,4740 = 7,957,920$ quadruples representing valid analogies. We trained our model on this augmented dataset.

Our first experiments enabled us to determine that one epoch took around 3 hours to run. We decided to train our model for 50 epochs, but we realised that the loss didn't decrease at the end of the training (it oscillated), which indicates that the model did not learn.

It could partly be explained by the fact that GloVe embeddings do not take morphology into account, but this explanation alone was not satisfactory. We thus decided to find a way to decrease the running time so that we could investigate the issues. We tried to use bigger data batches and to store the embedded quadruples in a file we would load instead of embedding each word during the training phase. However, none of these ideas significantly reduced the running time. We concluded that the size of the dataset was probably responsible for the time needed and decided to train a new model on $50,000$ analogies only (before augmentation) for 20 epochs. These training conditions are the same as for the embedding model described in Section 2.2.2, which is the one we used for all our other experiments.

As expected based on the results on the full dataset, the loss did not decrease on the smaller dataset. When we evaluated the model (see Section 2.3.1), the accuracy was of 100% which did not match the results from the loss. For this reason, we investigated the embeddings and noticed that many words were embedded with a vector full of zeros. After further analysis, we discovered that 49% of the words in the training set and 51% of the words in the test set were not present in the GloVe file we used to embed our words and were thus embedded by the default zero vector. Since an analogy is based on 4 words, it means that more than half of the analogies used for training contained at least one word embedded with a zero vector. This could explain why the model does not learn. Indeed, the model was trained on tuples $(embed(A), embed(B), embed(C))$ and was expected to produce $embed(D)$, where $A : B :: C : D$ holds true. But if one of the word was embedded with a zero vector, the analogy did not hold anymore, so the data we used for training the model contained invalid analogies that were (wrongfully) considered valid.

Moreover, if A, B, C and D are embedded as zero vectors, the model is taught that producing a zero vector is the right thing to do. Therefore, since this kind of data was probably fed many times to the model, it probably produced a zero vector all the time. As for the accuracy of 100%, we confirmed that all the expected and produced vectors were full of zeros.

After such results, we decided to abandon GloVe embeddings and started working on a character embedding model which is more relevant for a morphology task. At this stage, we had several options.

The first one consisted in training a character-level encoder trained with the regression model (Figure 1.2). The encoder would have embedded A, B, C and D separately and then the model would have run on these embeddings instead of the GloVe ones. However, this option has a major issue. The solution of the analogical equation $A : A :: A : X$ is always A, thus the model would always be correct if it learned a dummy embedding (the same embedding for all the possible inputs) and produced the same dummy embedding as a result.

The second option was also to create a character-level encoder, but trained on the classification task (Figure 1.1). Contrary to the regression task, negative examples are directly available for the classification task thanks to the permutation properties of analogies. These negative examples would force the model to learn real embeddings.

Eventually the last option was to create an auto-encoder trained separately. This model could work directly on the analogy solving task and would enable us to produce an actual word

(and not an embedding) as output of the regression model. However, the literature does not seem very developed for character level auto-encoders while we had some leads for an encoder. Hence, this task seemed too complicated for us. It still remains an interesting lead for future work.

2.2 Custom embedding model

Our aim was to solve morphological analogies. Our assumption was that we needed to investigate the structure of the words more than their meanings. For this reason, a character embedding model was more relevant than GloVe embeddings. Moreover, a character-level embedding model, once trained, is able to embed any word even those not encountered during the training phase, which solves the issue of the zero vector for unknown words (Vania, 2020).

2.2.1 Structure of our character-level embedding model

The structure described in Kim *et al.* (2016) is a CNN-LSTM language model where the CNN part embeds the words while the LSTM part investigates the relation between the words, *i.e.,* the context. As our aim was to embed words based on their morphological properties, the context was meaningless for our task. Our embedding model was thus inspired from the CNN part only.

Figure 2.1 describes the structure of our model. We first use a character embedding layer to encode each character of the word with a vector of size m. Characters never encountered during the training phase are embedded with vectors full of 0. At the beginning and at the end of the word, we add special vectors to signify the boundaries of the word. For a word of $|w|$ characters, we obtain a $(|w| + 2) \times m$ vector. We chose $m = 512$ for Japanese and $m = 64$ for the other languages. If we put Japanese aside, the biggest character dictionary contains 58 characters so we chose $m = 64$ because it is the closest power of 2 to $58 + 2$. Japanese's dictionary contains 632 characters so we chose $m = 512$.

Then the idea is to apply filters of different sizes on the embedding, each filter should recognize a pattern in the word: a filter of size 2 could recognize affixes such as "ab-", "be-", "in-" or "-en" in German for instance. We did not find literature on the maximal length of morphemes for our languages but we looked at an affix dictionary for German (IDS, 2018) in order to have an idea about it. Most of the affixes were of length 2 to 4 so we decided to use filters of size 2 to 6 to cover as many patterns as possible. We did not investigate the other languages but we could improve our model in the future with more information about the size of the morphemes. For each size we arbitrarily chose to use 16 filters. Further experiments are needed to determine the most efficient number of filters and the different sizes to use.

After the CNN layers, a max pooling layer is applied: we keep only the greatest number produced by each of the 80 filters (16 filters of 5 widths) so that only the most important patterns appear in the final embedding. We finally concatenate the results to produce an embedding of size 80.

2.2.2 Training phase and classification results

We decided to train this encoder with our classification model. Since we had both positive and negative data thanks to the properties of analogies (see Section 1.2), the encoder did not learn

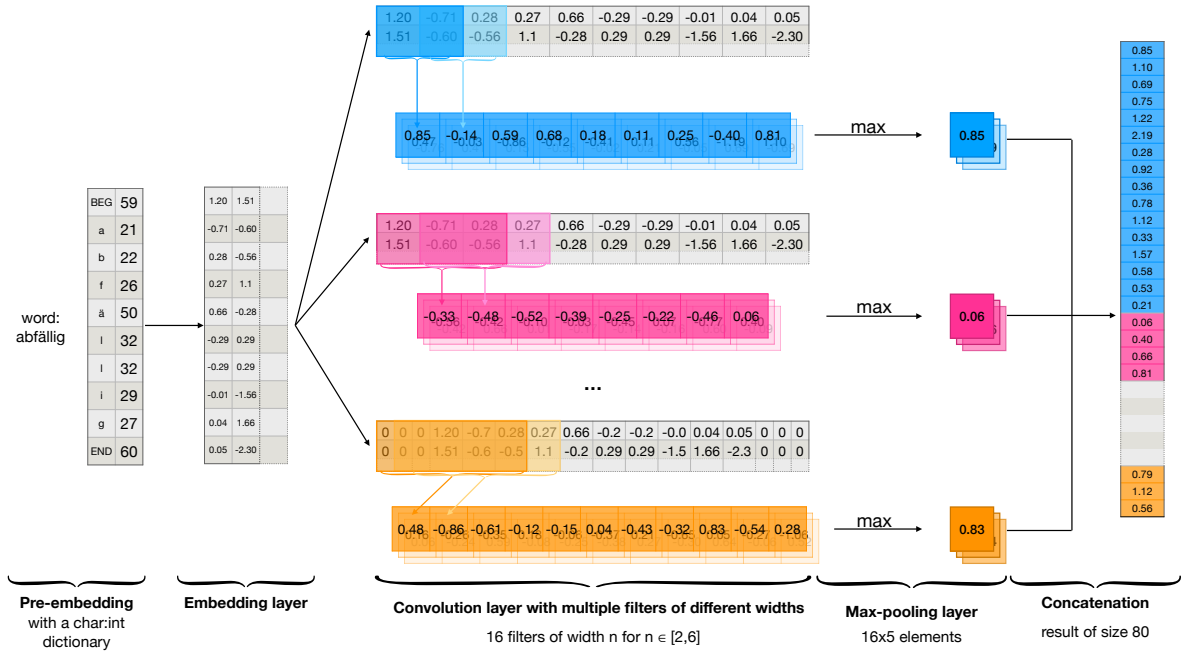


Figure 2.1: Structure of the CNN as a character level embedding model.

dummy embeddings as it could have done without negative data. Indeed with only positive data, learning the same embedding for all the words would make all the analogies look like $A : A :: A : A$, which is valid and the model would always be right. But if the model is trained on both positive and negative data, the previous dummy embeddings would lead it to classify invalid analogies as valid as the four elements would be equal. Using both positive and negative data forced it to learn meaningful embeddings.

We used the SIGMORPHON 2016 dataset for the training which enabled us to train 10 models (one for each language). Since the training and testing sets are distinct in this dataset, it is possible that the training file contains characters the testing file does not contain or the opposite (Table B.2). In the first case, it implies that many characters of the character to integer dictionary can have a different IDs during the training and the test part, so it yields poor results. We fixed this issue by using the dictionary produced for training as the dictionary for the test. However, this technique does not work in the opposite case, when the test file contains characters the training file does not contain.

A solution to both these issues would be using the UTF-8 codes as IDs, but the vocabulary would become very large as well as the embedding model since both are related. Another solution would be using an ID for the unknown characters, the same way context based word embeddings model use a vector for unknown words. Nevertheless, this solution would require that we retrain all of our models with one more character. To avoid this problem, we decided to use embeddings full of 0 for the unknown characters.

Most of the SIGMORPHON 2016 files yield several hundreds of thousands analogies and each analogy was used 11 times by the model (8 valid forms, 3 invalid). Using the entire datasets for the training would have taken a very long time, as it did with GloVe, so we decided to use only subsets of $50,000$ analogies and train for 20 epochs (which takes around 6 hours).

We also tried our model on Japanese, an ideographic language, to see if the results were similar. The classification task produced very good results (they are among the best ones) but the test set was small and even if the analogies of the test set were not used for training, they were based on the same words, while the test sets of other languages contain new words. For these reasons, Japanese could be less complex to deal with for the model.

The results of the classification task are described in Table 2.1. Most of the time, the classification of invalid analogies is less accurate than valid analogies. We thought about two major reasons for this. The first one is the presence of exceptions in the datasets (irregular verbs for instance). Such transformations are less likely to be shared by many words and are thus not recognised by our model. The second reason is the training setting. Since the model is trained on 3 invalid analogies and 8 valid analogies, *in practice* $150,000$ invalid analogies and $400,000$ valid ones in total, it is possible that the model needs to be trained on more invalid data to reach similar results as with valid data. We could apply the permutations properties on the 3 invalid analogies so that we would have 3×8 invalid analogies for 8 valid ones. However this could lead the model to focus more on invalid analogies and thus induce a drop in the accuracy for valid analogies. Moreover, we use valid and invalid data to make sure the embedding model does not learn dummy embeddings and an imbalance between the number of positive examples and the number of negative example could produce this result.

Examples for classification task on Arabic and German are shown in Table 2.2. Examples for all the languages are available in appendices (Tables C.1 and C.2).

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

Table 2.1: Accuracy results (in %) for the classification task. The Japanese model was trained on 18,487 analogies and tested on 7,923 of the same dataset (the two subsets were distinct). All the other models were trained on 50,000 analogies and tested on 50,000 of different sets except for Maltese, Navajo and Turkish which were tested on 3,707, 4,843 and 11,360 analogies due to the size of the dataset.

No.	Lang.	Expected	Result	Analogy	Form
1	ARA	valid	valid	naffaqa:naffaqnā::dammama:dammamnā	A:B::C:D
2	ARA	valid	invalid	bayyāᶜūna:al-bayyāᶜu::nawarun:an-nawariyyu	B:A::D:C
3	ARA	invalid	invalid	dammama:naffaqnā::naffaqa:dammamnā	C:B::A:D
4	ARA	invalid	valid	al-ʾamtiᶜatu:al-matāᶜu::al-qīmatu:al-qiyamu	B:A::C:D
5	GE	valid	valid	extrovertiert:extrovertiertere::angelsächsisch:angelsächsischere	A:B::C:D
6	GE	valid	invalid	entgehen:entgingen::schwächen:schwächten	A:B::C:D
7	GE	invalid	invalid	extrovertiertere:extrovertiert::angelsächsisch:angelsächsischere	B:A::C:D
8	GE	invalid	valid	unkommunikative:unkommunikativ::abgestrahlt:abgestrahlte	B:A::C:D

Table 2.2: Classification examples of Arabic (ARA) and German (GE). We call the words A, B, C, and D depending on the order they appear in the dataset. A always forms a pair with B and C always forms a pair with D.
For Arabic, in example (2), the reason the model fails might be due to the presence of a dicrtic character in B that the model interprets as an affix. The model, therefore, reads the transformation between B and D as not corresponding to one another. For example (4), the analogy is invalid because of its form but the model fails to notice it, which may be related to the smaller amount of negative data during the training compared to positive data.
For German, example (6) uses an irregular verb, which probably explains why the model fails. As for example (8), the analogy is invalid because of its form, like example (4), which the model also fails to notice.

2.3 Solving analogies

The classification task enabled us to train embedding models which hopefully represent accurately the morphology of the words. This leads us to the analogy solving task. Indeed, for an analogical equation $A : B :: C : X$, the embeddings must encode the morphological features of A, B and C so that the relation between A and B is also to be found between $embed(A)$ and $embed(B)$. As explained in Section 1.1.2, we use the neural network proposed by Lim et al. (2019) to solve this task.

The main issue with our approach is that we use a CNN encoder and not an auto-encoder. It means that when the neural network for regression produces a vector, we have no tool to transform this vector into a word. In order to match the produced vectors with actual words, we considered all the embeddings of the words in the dataset for a given language and assumed the closest vector to the produced one would be chosen as output. This method raises several questions especially regarding the metrics we use and the way to deal with words equally close to each others.

Our aim was to compare our results to those of Murena *et al.* (2020) who proposed an empirical approach to solve morphological analogies which produced competitive results.

2.3.1 Evaluation method

Before training our models, we had to decide on an evaluation method. As mentioned before, we worked with a word encoder and thus obtained vectors we cannot decode as output from the regression neural network. These vectors were not equal to the expected ones so comparing the expected vectors to the produced ones gave an accuracy of 0%. However, if we choose a method to match a produced vector with one corresponding to the embedding of a word of the dataset and this vector corresponds to the expected ones, we can consider the produced vector as right.

Given a produced vector, the idea was to find the closest one in a set of known embeddings. To do this, we first stored the embeddings of all the words in the dataset of a given language, *e.g.* the words of the testing set as well as those of the training set. As we evaluated our model on the testing set, we did not need the embeddings of the words of the training set. However, if we search among a bigger set, the accuracy we compute is more meaningful as it reduces the possibility to get the right vector out of luck.

Then we had to choose a (dis)similarity metrics. The Cosine similarity and the Euclidean distance given by Equation (2.1) and Equation (2.2), respectively, are the most commonly used:

$$Cosine\_similarity(X, Y) = \frac{X \cdot Y}{\|X\|\|Y\|} \tag{2.1}$$

$$Euclidean\_distance(X, Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.2}$$

where $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_n)$. Hence we decided to use both and compare the results. More precisely, given a produced vector X, we computed its Cosine similarity (resp. Euclidean distance) with all the stored embeddings and retrieved the vector Y such that maximizes $Cosine\_similarity(X, Y)$ (resp. minimizes $Euclidean\_distance(X, Y)$).

This method enabled us to match each produced vector with one belonging to the stored embeddings. As the expected tensors are embeddings of words present in this set as well, if the produced vector is the right one then the one closest to it should be exactly the expected

one. However, the process we used to store and then load the embeddings slightly modified the tensors values: from the fifth decimal, the values of the components of the produced tensors differ from the values of the components of the stored tensors. To tackle this issue, we considered that the expected vector and the closest one were the same if their components were equal two by two up to the fourth decimal.

With this method, there was a possibility that several vectors were equally close to the produced one. During our first evaluation, we decided to use the first found vector with the biggest similarity (resp. smallest distance). Later, we decided to check if the expected vector was slightly further and thus considered all the vectors in a given similarity (resp. distance) range.

2.3.2 Results

We trained the models for each language separately. For all of them, the training set consisted in 50,000 analogies. The four words were embedded thanks to the trained CNN neural network corresponding to the language of the data. Then for each quadruple of vectors, we used the eight valid permutations as input data: for a valid permutation of the form $A : B :: C : D$, we applied the regression model on $(embed(A), embed(B), embed(C))$ and compared the result to $embed(D)$ with mean squared error before the back-propagation. In the end, the model was trained on 400,000 analogies.

2.3.2.1 First evaluation

As explained before, the evaluation method was such that the produced vector could be matched with several vectors of the set of stored embeddings. For our first evaluation, we compared the expected vectors only with the first matching vector among the stored ones (*e.g.* the one with the smallest identifier). The results are described in Table 2.3, the last column contains the results of (Murena *et al.*, 2020) for the same task.

Japanese produced the worst results which could be explained by the length of the dataset as well as the distribution of the embeddings. Also, our results are not as good as those of (Murena *et al.*, 2020) on all the languages. However, the language on which it performed best is Georgian with both approaches, and the two worst languages (apart from Japanese) are also the same. This indicated that our approach was relevant.

2.3.2.2 Further evaluation

Our results for solving analogies were far from the ones of (Murena *et al.*, 2020) for most languages but they were encouraging. Thus we wanted to know if the right vector was far from the predicted one when our model failed. If not, it would indicate our neural approach is relevant and could achieve better results with more training and/or fine-tuning. To do this, we implemented a new evaluation method.

1. Given A, B, and C our model produces $D_{predicted}$;
2. We then compute the Cosine similarity (resp. Euclidean distance) between $D_{predicted}$ and all the stored embeddings;
3. We order the stored embeddings according to their Cosine similarity (resp. Euclidean distance) with $D_{predicted}$ so that the vector of rank 1 is the closest to $D_{predicted}$;

Language	Cosine similarity	Euclidean distance	(Murena *et al.*, 2020)
Arabic	51.41	51.53	87.18
Finnish	72.84	72.23	93.69
Georgian	93.37	93.44	99.35
German	87.55	87.78	98.84
Hungarian	68.31	68.13	95.71
Japanese	19.76	17.50	/
Maltese	75.68	76.94	96.38
Navajo	45.81	47.41	81.21
Russian	69.57	69.05	96.41
Spanish	87.86	87.53	96.73
Turkish	70.10	67.77	89.45

Table 2.3: Accuracy results (in %) for the analogy solving task. The Japanese model was trained on $18,487$ analogies and tested on $7,923$ of the same dataset (the two subsets were distinct). All the other models were trained on $50,000$ analogies and tested on $50,000$ of different sets except for Maltese, Navajo and Turkish which were tested on $3,707$, $4,843$ and $11,360$ analogies due to the size of the dataset.

4. We retrieve the rank 1 vector $D_{closest}$;
5. We retrieve the next vectors in the ordered list until the Cosine similarity (resp. Euclidean distance) is too small (resp. too large) compared to the one of the vector of rank 1, we use a parameter k to make the maximal difference vary;
6. If the expected vector is among the retrieved ones then we consider the model was right and we retrieve the rank of the right vector for statistics;
7. If the expected vector is not among the retrieved ones then we consider our model failed and add a 0 to the list of ranks.

In practice, if $Cosine\_similarity(D_{predicted}, D_{closest}) = s_0$, we retrieved all the vectors t such that $Cosine\_similarity(D_{predicted}, t) \leq (1-k) * s_0$ for $k \in \{0, 0.01, 0.02, 0.05\}$. We used $k = 0$ to see if there were often several vectors which shared the same similarity (resp. distance) from the predicted one, which is not the case (the length of the sets of retrieved vectors are available in appendices: Table D.1). Table 2.4 provides a few examples of the words produced by the model for $k = 0.02$. Examples for all the languages are available in appendices (see Tables C.3 to C.5).

Several factors influence why the Arabic model would fail. Even when the size is set, the variety of affixes in Arabic could affect the validity of the analogy. Both pairs should follow the same pattern (have the same format and use the same affix) for the model to be valid. Another factor is the fact that the model might interpret that some words contain an affix, when, in fact, they don't (for instance the word "determine" seems to contain the affix "de-"). As for the third Arabic example, we aren't sure why the model failed. The same words were used in the previous examples but presented in a different order (words belonging to the same pair are placed next to one another) and the model was able to find the expected result. Therefore, we aren't able to provide an explanation as to why the model failed to find the expected result for the third example.

	Analogical equation	Expected	Rank
Arabic	al-ʾabzanu:al-ʾabzanayni::az-zaᶜīmu::?	az-zaᶜīmayni	1
	$\xrightarrow[k=0.02]{Results}$ **az-zaᶜīmayni**, al-māᶜizayni, al-māᶜizatayni, al-maᶜzatayni, al-qirmiziyyayni, al-quzīmiyyayni, al-qamīṣayni, al-qirmiziyyatayni, al-quzīmiyyatayni		
	al-ʾabzanayni:al-ʾabzanu::az-zaᶜīmayni::?	az-zaᶜīmu	4
	$\xrightarrow[k=0.02]{Results}$ al-maᶜzatu, al-māᶜizatu, al-māᶜizu, **az-zaᶜīmu**, al-mawāᶜizu, al-qirmiziyyatu, al-maᶜlūmu		
	az-zaᶜīmayni:az-zaᶜīmu::al-ʾabzanayni::?	al-ʾabzanu	Not found
	$\xrightarrow[k=0.02]{Results}$ al-juzayʾu, al-ʾiblīsu, al-ʾaṯlāṯu, al-baǩšīšu, al-juzʾu		
German	toxisch:populär::toxischere::?	populärere	1
	$\xrightarrow[k=0.02]{Results}$ **populärere**, populäre		
	ausgestorben:ausgestorbenes::mitverantwortlich::?	mitverantwortliches	2
	$\xrightarrow[k=0.02]{Results}$ mitverantwortlicher, **mitverantwortliches**, mitverantwortliche		
	applizierten:applizieren::versähen::?	versehen	Not found
	$\xrightarrow[k=0.02]{Results}$ versähen, verschlingen, verschieben		

Table 2.4: Examples for Arabic and German for solving analogies.

In the first German example, the right vector is the one closest to the vector produced by the model. In the second example however, the right vector is the second one. When we look at the words retrieved for $k = 2$, we can see that they are similar (they differ on their last character only) which could explain that our model was not completely accurate. Eventually the third German example uses an irregular verb, which probably explains why the model fails.

During these evaluations, we computed the *mean reciprocal rank* (MRR) given by

$$\text{MRR} = \frac{1}{n}\sum_{i}^{n}\frac{1}{rank_i} \quad (n \text{ the number of evaluations})$$

as well as the mean rank of the right vector when it is greater than one. The MRR evaluates the rank of the correct vector among the set of found ones, if the correct one is not in the set then the reciprocal rank for this evaluation is 0. Thus a value close to 1 means the right vector is often close to the predicted one while a value close to 0 means the right vector is very far from the predicted one.

The results for $k \in 0.01, 0.02, 0.05$ are described in Table 2.5. As we could expect, the accuracy rises when we increase the search area. The results are more striking for Cosine similarity than Euclidean distance because we discover more vectors when we slightly decrease the Cosine similarity than when we increase the Euclidean distance from the same percentage due to the definition of these metrics. For this reason from now on we will only refer to the results with Cosine similarity (the full results are available in appendices: Tables D.2 and D.3). With this metric, our model outperforms (Murena *et al.*, 2020) for some languages.

Statistics about the mean rank of the right vector, available in Table 2.6, show that most of the time, the rank of the right vector is not very high. For some languages the mean rank for

	Cosine similarity				Euclidean distance			
	$k = 0$	$k = 0.01$	$k = 0.02$	$k = 0.05$	$k = 0$	$k = 0.01$	$k = 0.02$	$k = 0.05$
Arabic	51.41	73.06	85.29	97.19	51.53	54.49	57.35	65.40
Finnish	72.84	90.47	95.92	99.06	72.21	74.10	75.78	80.33
Georgian	93.37	96.98	97.77	98.65	93.44	93.88	94.27	95.24
German	87.55	93.00	95.49	98.38	87.78	88.51	89.16	90.92
Hungarian	68.31	86.75	93.36	98.32	68.13	69.97	71.79	76.90
Japanese	19.76	68.37	78.09	93.04	17.50	19.42	21.51	27.93
Maltese	75.68	84.10	86.96	91.26	76.94	78.03	79.26	82.68
Navajo	45.81	57.52	67.25	85.07	47.41	49.67	51.92	58.55
Russian	69.56	77.58	83.89	94.27	69.04	70.44	71.86	75.62
Spanish	87.86	96.05	97.88	99.24	87.53	88.62	89.59	92.01
Turkish	70.10	80.64	86.61	94.72	67.77	69.58	71.26	76.03

Table 2.5: Accuracy (in %) for the analogy regression task when searching for the expected vector in a wider range.

$k = 0.05$ is much higher than for other values but the standard deviation shows a high variability in these cases. We confirmed with boxplots (Figure D.1) that these higher values were due to outliers.

These results reinforce the idea that our neural network approach is suitable and it is possible that with further work on the embedding model our approach would give similar results when considering only the closest vector as a valid candidate. Developing an auto-encoder, and thereby fixing all the issues related to the evaluation method is all the more an interesting lead.

	MRR			MR±sd when rank>1		
	$k = 0.01$	$k = 0.02$	$k = 0.05$	$k = 0.01$	$k = 0.02$	$k = 0.05$
Arabic	0.60	0.63	0.64	2.9±1.6	4.4±4.3	12.2±28.4
Finnish	0.80	0.81	0.81	3.1±2.0	4.6±5.2	9.8±26.2
Georgian	0.95	0.95	0.96	2.1±0.5	2.4±1.2	4.2±8.2
German	0.90	0.91	0.92	2.3±0.7	2.7±1.8	5.3±12.1
Hungarian	0.76	0.77	0.78	2.9±1.4	3.4±2.1	4.1±4.0
Japanese	0.28	0.30	0.30	32.3±53.8	38.4±77.3	39.7±72.9
Maltese	0.79	0.79	0.80	4.6±6.2	8.0±21.4	17.8±124.5
Navajo	0.51	0.54	0.58	2.4±0.9	3.0±1.7	5.3±7.4
Russian	0.73	0.76	0.80	2.2±0.5	2.4±1.0	3.3±4.1
Spanish	0.92	0.92	0.92	2.5±1.1	3.0±2.7	5.3±14.2
Turkish	0.75	0.77	0.79	2.3±0.8	2.7±1.4	4.4±5.7

Table 2.6: Mean reciprocal rank of the right vector and mean rank ± standard deviation when the rank is higher than 1.

Chapter 3

Transfer learning

As explained before, we trained two models during the classification task: the embedding model and the classification model. These models are dependent on the language that they were trained on. For instance, the dictionary of characters depends on the language, and they are also interdependent since they are trained together. We applied the different models trained on each language to the other languages of the dataset. The objective is to explore the generalization capabilities of the CNN model, and to test its dependence on the training language.

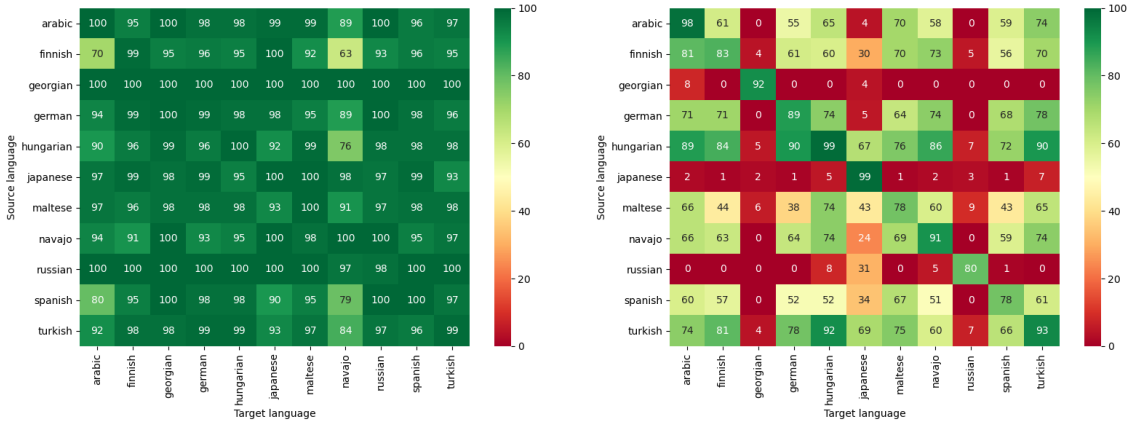
We wanted to see how these models could transfer from one language to another. To evaluate their generalization capacity, we ran the evaluation using each model on all the languages. In Section 3.1 we describe the results when transferring both the embedding and classification models, which we called full transfer. Then in Section 3.2 we transfer only the classification model (the data is embedded with the "right" embedding model), which we called partial transfer.

3.1 Full transfer

The results for full transfer on positive and negative data are presented in Figure 3.1. In this experiment, the embedding model and the classifier were congruent, *i.e.* they were trained together. For positive data the results are most of the time above 90% except for Arabic and Navajo words. They are more heterogeneous for negative data. The main reason is probably the character dictionary gap between the language of the models and the language of the data. Characters unknown to the model are indeed embedded as zeros. If the data contains mainly unknown characters, the embeddings do not really reflect the words.

Accuracy close to 100% for positive data with accuracy close to 0% for negative data is most likely the result of almost unrecognized languages. Indeed, in this case most of the words would be embedded as ε and the analogies would look like $\varepsilon : \varepsilon :: \varepsilon : \varepsilon$ whether in positive or negative form, so the positive analogies would rightly be classified as valid (accuracy of 100%) while the invalid ones would falsely be classified as valid (accuracy of 0%).

The Hungarian models seem interesting if we want to work with analogies of several languages at the same time.



(a) Valid analogies

(b) Invalid analogies

The source language is the language of the models, the target language the language of the data.

Figure 3.1: Accuracy for full transfer of the models.

3.2 Partial transfer

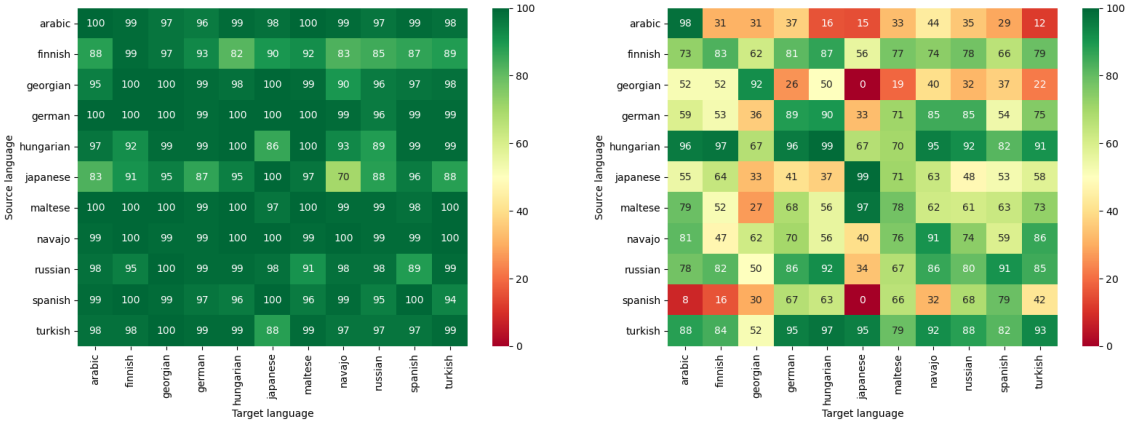
To solve the issues we had with character dictionaries for full transfer, we tried to transfer only the classifier. However, it induced that the embedding model and classification model were no more congruent. The results are displayed in Figure 3.2. As for the full transfer, overall results are good even though they remain heterogeneous for negative data.

As we used the embedding model corresponding to the language, we have less combinations producing an accuracy of 0% for negative data. Only Georgian *to* Japanese, and Spanish *to* Arabic and Japanese produce this result.

3.3 Discussion

The first observation we can make is the non-symmetry of the diagrams: the results are not necessarily similar when we transfer *from* a given language to the others to the results when we transfer from other languages *to* that language. For instance the Hungarian model transfers really well to all the other languages while most of the other models are not that efficient with Hungarian (which is even more the case for partial transfer than full transfer). The statistics we computed on the different languages do not indicate particularities of Hungarian compared to the other languages. However, Hungarian has a "particularly rich morphology" (Kiefer, 2010) using inflection, derivation and compounding. We have no proof that it influences the transferability of the model but further investigations about the different morphological transformations used by the languages could improve our understanding of these results.

Then we can see that the Arabic model is efficient for full transfer but not really for partial transfer. This may be due to the fact that Arabic words are formed by roots and word patterns, and the use of affixes is rather limited and slightly different compared to other languages. It is thus possible that the Arabic embedding model encodes the sub-words differently from the other models. In this case the embedding and classification model would be strongly related, which could result in a poor performance once applied to other languages in case of partial transfer. As Maltese is also a Semitic language (Harwood, 2021), we could expect that using the Arabic model on Maltese in partial transfer would be more efficient than the results with negative



(a) Valid analogies

(b) Invalid analogies

The source language is the language of the classification model, the target language is the language of the data and of the embedding model.

Figure 3.2: Accuracy for partial transfer of the models.

data indicate. However, Maltese's morphology is different from other Semitic languages, where it has been influenced by Sicilian, Italian, French and more lately English (Mifsud, 2017) and would have a "general tendency [...] to exhibit productive concatenation, rather than root-and-pattern, word formation processes" (Perea *et al.*, 2012). Though it has some shared words with Arabic and they appear to have a similar form of construction and pattern, the influence of other languages can't be ignored. Thus the fact that the Arabic model is not very efficient with Maltese in partial transfer support the idea that the Arabic embedding model encodes the morphology of words differently than most models. The issue could be as well that Maltese encoding model might have encoded the morphology of Maltese words differently due to the reasons mentioned above, which made the classifier unable to produce the expected results.

Surprisingly, the accuracy is not always of 0% for negative data when the models transfer to Japanese. None of the Japanese characters are present in the dictionaries of the other models so we could expect all the analogies to look like $\varepsilon : \varepsilon :: \varepsilon : \varepsilon$ and thus all the said invalid ones to be classified as valid.

Eventually Hungarian and Russian seem to be very efficient models in terms of transfer learning. Because of the alphabet gap, the Russian model performs poorly on negative data in full transfer but the results in partial transfer are above average. In average, Hungarian is one of the languages closest to all the others in the family hierarchy, which may explain why it transfers well. Russian however is not particularly close to the rest of the languages, further experimentation and research on the morphology of Russian are needed to explain this result.

Chapter 4

Conclusion and perspectives

This report provides a detailed summary of the work carried for the realization part of the project. As discussed, different approaches are used to tackle each type of analogy. Though many articles have adopted different approaches to solve semantic analogies including Textual Analogy Parsing (TAP), Dependency Relations (DR), Vector Space Model (VSM) and learned Neural Network analogy, we noticed that not many articles tried to tackle morphological analogies, which are the focus of our project. Therefore, we worked on analyzing morphological analogies by adapting a novel approach. Instead of using the already existing approaches to analyze morphological analogies like Kolmogorov complexity and CopyCat, we developed a neural approach to identify and complete morphological analogies.

We used two datasets which were introduced in section 1.3 to import 11 languages. We inspired our work from the approach of (Lim et al., 2019), where we successfully adapted their neural network approach to solve morphological analogies thanks to our custom embedding model. With our approach, we managed to achieve competitive results to those of (Murena *et al.*, 2020). Compared to the model of Lim *et al.*, our CNN model is more flexible in many terms:

- it is able to carry over domain and language specificities from the training process;
- it is able to model any words even those never encountered in the training phase;
- it has strong potential to carry over models of analogy when using an adapted embedding model as shown with our transfer experiments.

The word embedding model is a pillar for both tasks and improving it would most likely induce better results for both classification and analogy solving. In this chapter we mention some feasible directions that we will pursue to improve our models.

4.1 Improvement of the training settings

During the training of the embedding and classification models, we used some properties of analogies to augment our dataset: given one valid analogies we generated in total 8 valid ones and 3 invalid ones. We could apply the augmentation to the 3 invalid analogies and thereby obtain 24 invalid analogies for 8 valid ones in the end. This could help improving the results for negative data.

Moreover, we chose to train our models on 50,000 analogies but we have more available

(Japanese aside). It could be interesting to see if more data would improve the results, in particular for languages with more different transformations and less word pairs per transformation.

Eventually, we did not dedicate an ID to the unknown characters when we trained the models. They are thus embedded with vectors full of zeros and the model most likely ignores them. Retraining the models with an additional ID for unknown characters could improve our results, in particular for full transfer.

4.2 Towards a multilingual word embedding model?

We trained separately one model per language but we could imagine training a model on several languages at the same time. The embedding model would learn more characters and more patterns which could give it more robustness for transfer learning. The chosen languages should be such that they use similar alphabets (we should not use Russian or Georgian with German for instance) and the coverage of the union of their character dictionaries should be as high as possible on languages using a similar alphabet. Eventually, choosing languages as far away as possible from each other in terms of phylogeny could provide more difference in the morphology. It is possible that such a model would also tackle the analogy solving task on several languages.

4.3 Discussion about ideographic languages

As we experimented with Japanese only, we do not have much material to make assumptions on the usability of our approach for ideographic languages. However, we can suppose that the size of the character dictionary plays a role in the quality of the embedding model.

We cannot explain why the classification task performs well while the analogy solving task does not for Japanese. It could be related to the size of the dataset: classifying Japanese analogies could require less learning data than solving them. However, we tested the analogy solving task on Japanese with other models and the accuracy was close to 0. As we did not try to transfer the models on other languages, we cannot say if it is related to the Japanese language itself or if our regression models do not support transfer at all. Further experimentation in this direction would be interesting.

4.4 Final remarks

Our early experiments on transferability highlighted the potential to transfer and reuse our neural approach across domains. The results also confirmed our hypothesis that morphological analogy models are transferable between similar languages (in terms of alphabet and morphology). Further results on transferability could be used to measure morphological similarities between languages.

However our embedding model is dependant on the characters encountered during the training. Its embedding layer encodes the unknown characters with zeros and we do not know if the model then ignores them or is able to consider them as a part of the sub-words and morphemes. This "black-box" effect is a drawback of the neural network approach and it would be interesting to have a better understanding what our three models really do. Interpretability is indeed more and more discussed and literature propose various methods which could help us understanding the role of each feature (in our case characters) in the predicted results (Lundberg & Lee, 2017).

For instance Saliency Maps (Salehi, 2020) could help us highlighting which parts of the words are the most used in practice by our embedding model as well as our classifier model and also if the relation between the two pairs is more or less important than the relations between the words of the pairs for the latter.

This project required knowledge in Neural Networks as we implemented the classification model as well as the CNN embedding model. When we started the realisation part we had almost none about this subject but our supervisor, Esteban Marquer taught us a lot. Several courses during the semester enabled us to have a better understanding of Neural Networks and this project was the opportunity to apply our knowledge on a real life problem. The morphology course in the first semester gave us some basis to understand morphological analogies and more specifically the different transformations words can undergo but more knowledge on the similarities between languages could have helped us interpreting our results. This project was also the opportunity for us to take part in the writing of a paper together with our supervisors, Miguel Couceiro and Esteban Marquer, as well as Pierre-Alexandre Murena, one of the authors of (Murena *et al.*, 2020). This paper was submitted to the IEEE International Conference on Data Science and Advanced Analytics 2021.

Bibliography

Balouek *et al.* (2013). Adding virtualization capabilities to the Grid'5000 testbed. In I. I. Ivanov, M. van Sinderen, F. Leymann, & T. Shan (Eds.) *Cloud Computing and Services Science*, vol. 367 of *Communications in Computer and Information Science*, (pp. 3–20). Springer International Publishing.
URL https://hal.inria.fr/hal-00946971

Betrand (2016). Types of analogies.
URL https://magoosh.com/mat/types-of-analogies-on-the-mat/

Chiu, A., Poupart, P., & DiMarco, C. (2007). Generating lexical analogies using dependency relations. In *the Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, (pp. 561–570).
URL https://www.researchgate.net/publication/221012942_Generating_Lexical_Analogies_Using_Dependency_Relations

Cole, T., & Siebert-Cole, E. (2020). Trees of languages.
URL https://www.researchgate.net/publication/344272626_Family_Trees_of_Languages_-_poster_titles_and_languages_with_links

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Mach. Learn.*, *20*, 273–297.
URL https://doi.org/10.1023/A:1022627411411

Cotterell *et al.* (2016). The sigmorphon 2016 shared task—morphological reinflection. In *the Proceedings of the ACL 2016 Meeting of SIGMORPHON*, (pp. 10–22).
URL https://www.aclweb.org/anthology/W16-2002

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*, (pp. 1–27).
URL https://www.aclweb.org/anthology/K18-3001

Couceiro *et al.* (2017). Analogy-preserving functions: A way to extend boolean samples. In *the Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, (pp. 1575–1581).
URL https://www.ijcai.org/proceedings/2017/0218.pdf

Delhay, A., & Miclet, L. (2004). Analogical equations in sequences: Definition and resolution. In *the Proceedings of Springer on Grammatical Inference: Algorithms and Applications*, (pp. 127–138).
URL https://link.springer.com/chapter/10.1007/978-3-540-30195-0_12

Gladkova, A., Drozd, A., & Matsuoka, S. (2016). Analogy-based detection of morphological and semantic relations with word embeddings: what works and what doesn't. In *the Proceedings of the NAACL Student Research Workshop*, (pp. 8–15).
URL https://www.aclweb.org/anthology/N16-2002

Grave *et al.* (2018). Learning word vectors for 157 languages. In *the Proceedings of the International Conference on Language Resources and Evaluation, (LREC 2018)*.
URL https://www.aclweb.org/anthology/L18-1550

Harwood, M. (2021). Europeanization and language: the impact of eu language status on maltese. *Journal of contemporary European studies*, *ahead-of-print*, 1–14.
URL https://doi.org/10.1080/14782804.2021.1884534

Haspelmath, M. (2002). *Understanding morphology*. Arnold, London.
URL https://doi.org/10.5281/zenodo.1236482

IDS (2018). Institut für Deutsche Sprache: "Wörterbuch der Affixe". Grammatisches Informationssystem grammis.
URL https://grammis.ids-mannheim.de/affixe

Karpinska *et al.* (2018). Subcharacter Information in Japanese Embeddings: When Is It Worth It? In *the Proceedings of the ACL Workshop on the Relevance of Linguistic Structure in Neural Architectures for NLP*, (pp. 28–37).
URL http://aclweb.org/anthology/W18-2905

Kiefer, F. (2010). Hungarian. *Revue belge de philologie et d'histoire, tome 88, fasc. 3, 2010. Langues et littératures modernes.*.
URL https://www.persee.fr/doc/rbph_0035-0818_2010_num_88_3_7801

Kim *et al.* (2016). Character-aware neural language models. (p. 2741–2749).
URL https://arxiv.org/abs/1508.06615

Lamm *et al.* (2018). Textual analogy parsing: What's shared and what's compared among analogous facts.
URL https://www.aclweb.org/anthology/D18-1008

Lepage, Y. (2003). *De l'analogie rendant compte de la commutation en linguistique*. Habilitation 'a diriger des recherches, Universit'e Joseph-Fourier - Grenoble I.
URL https://tel.archives-ouvertes.fr/tel-00004372

Levenshtein, V. (1965). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics. Doklady*, *10*, 707–710.
URL https://www.bibsonomy.org/bibtex/220546d80ce76f58c6ef6ece9dd5f5056/jimregan

Lim, S., Prade, H., & Richard, G. (2019). Solving word analogies: A machine learning perspective. In G. Kern-Isberner, & Z. Ognjanovic (Eds.) *the Proceedings of the 5th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU 2019*, vol. 11726, (pp. 238–250).
URL https://doi.org/10.1007/978-3-030-29765-7_20

Lundberg, S., & Lee, S.-I. (2017). A unified approach to interpreting model predictions.
URL https://www.bibsonomy.org/bibtex/2224af0bb544c9baca23c29a903d6a264/jakobgerst

Miclet, L., Bayoudh, S., & Delhay, A. (2008). Analogical dissimilarity: Definition, algorithms and two experiments in machine learning. *Journal of Artificial Intelligence Research*, *32*, 793–824.
URL http://dx.doi.org/10.1613/jair.2519

Miclet, L., & Delhay, A. (2003). Analogy on Sequences : a Definition and an Algorithm. Research Report RR-4969, INRIA.
URL https://hal.inria.fr/inria-00071610/file/RR-4969.pdf

Mifsud, M. (2017). *Loan verbs in Maltese: A descriptive and comparative study*. BRILL.
URL https://brill.com/view/title/1643

Murena *et al.* (2020). Solving analogies on words based on minimal complexity transformation. In *the Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, (pp. 1848–1854).
URL https://www.ijcai.org/Proceedings/2020/0256.pdf

Paszke et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In *the Proceedings of Neurips on Advances in Neural Information Processing Systems 32*, (pp. 8024–8035).
URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In *the Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*, (pp. 1532–1543).
URL http://www.aclweb.org/anthology/D14-1162

Perea *et al.* (2012). Are all semitic languages immune to letter transpositions? the case of maltese. *Psychonomic bulletin & review*, *19*(5), 942–947.
URL https://doi.org/10.3758/s13423-012-0273-3

Salehi, M. (2020). A review of different interpretation methods (part 1: Saliency map, cam, grad-cam).
URL https://mrsalehi.medium.com/a-review-of-different-interpretation-methods-in-deep-learning-part-1-saliency-map-cam-grad-cam-3a34476bc24d

Turney, P. D. (2001). Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *the Proceedings of ECML on Machine Learning, ECML 2001*, (pp. 491–502).
URL https://arxiv.org/pdf/cs/0212033.pdf

Turney, P. D. (2006). Similarity of semantic relations. *Computational Linguistics*, *32*, 379–416.
URL http://dx.doi.org/10.1162/coli.2006.32.3.379

Van Rossum, G., & Drake, F. L. (2009). *Python 3 Reference Manual*.
URL https://docs.python.org/3/reference/

Vania, C. (2020). *On Understanding Character-level Models for Representing Morphology*. Ph.D. thesis, University of Edinburgh.
URL http://dx.doi.org/10.7488/era/49

Appendix A

Glossary

Acronym	Full name
ACC	Accusative
ACT	American College Test
ADJ	Adjective
CNN	Convolutional neural network
DAT	Dative
DP	Dependency Relation
FEM	Feminine
gen	Gender
ID	Identifier
Lang.	Language
LSTM	Long Short-Term Memory
MRR	Mean reciprocal rank
MSD	Morphosyntactic descriptions
N	Noun
NN	Neural Network
num	Number
PL	Plural
pos	Part of speech
SAT	Standardised Assessment Test
SG	Singular
TAP	Textual Analogy Parsing
TOEFEL	Test of English as a Foreign Language
VSM	Vector Space Model
UTF-8	Unicode Transformation Format−8-bit

Table A.1: Acronyms used in the report.

Parameter	Explanation
$D_{predicted}$	The vector produced by the analogy solver model
$D_{closest}$	The closest vector to $D_{predicted}$ in the set of stored embeddings
k	The extra portion of distance allowed to find the right vector in the solving task

Table A.2: Parameters used in the report.

Appendix B

Statistics on the datasets

In this Appendix we propose different statistics on the datasets which show the (dis)similarities between the languages we worked with.

Table B.1 presents the mean length (± standard deviation) of the words of the training sets, the number of different words, the number of different transformations, the mean number of pairs of words per transformation (± standard deviation) and the mean Levenshtein distance (± standard deviation).

Table B.2 provides a few comparative statistics between the training and test sets. It contains the portion of new words in the test sets compared to the training sets, the number of new transformations, the number of different characters in both sets and the number of new characters in the test sets compared to the training sets.

Appendix B provides more information about the length of the words in the training and test sets thanks to boxplots.

Figure B.2 provides more information about the number of words per transformation in the training and test sets thanks to boxplots.

Figure B.3 provides more information about the Levenshtein distance in the training sets thanks to boxplots (the values were similar for the test sets).

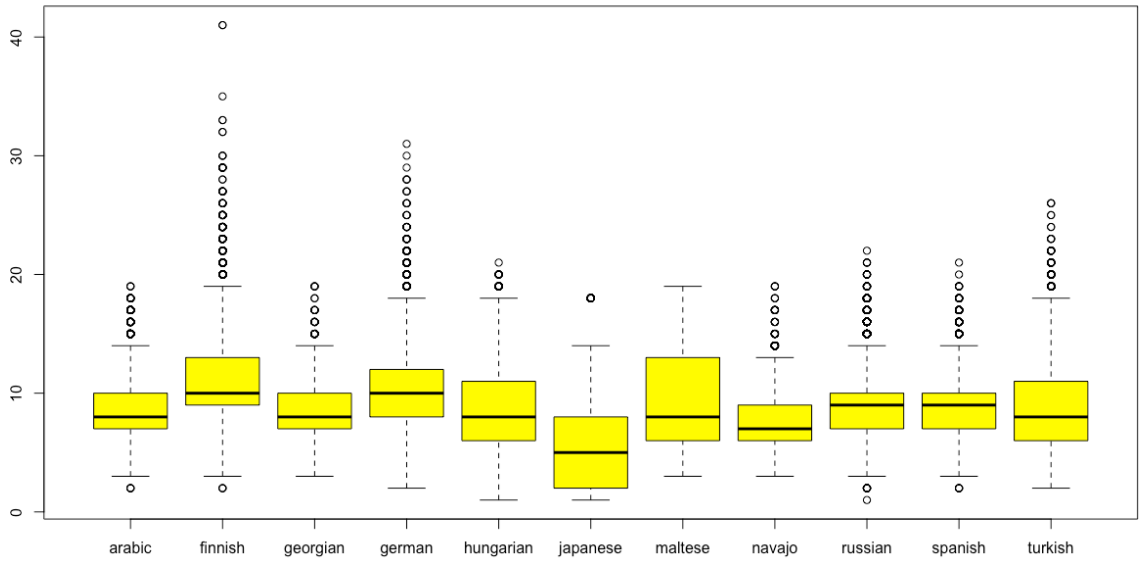
Figure B.4 indicates the coverage of the character dictionaries of the training sets on the character dictionaries of the test sets. For instance a coverage of 100% indicates that all the characters of the test set are present in the training set while a coverage of 0% indicates that all the characters of the test set are new. The last column indicates the mean coverage of each training set. The results presented on Figure B.4 are related to those of full transfer for the classification task as the embedding models know only the characters present in the set they were trained on.

Table B.1: Statistics on the word and word pairs of the training sets

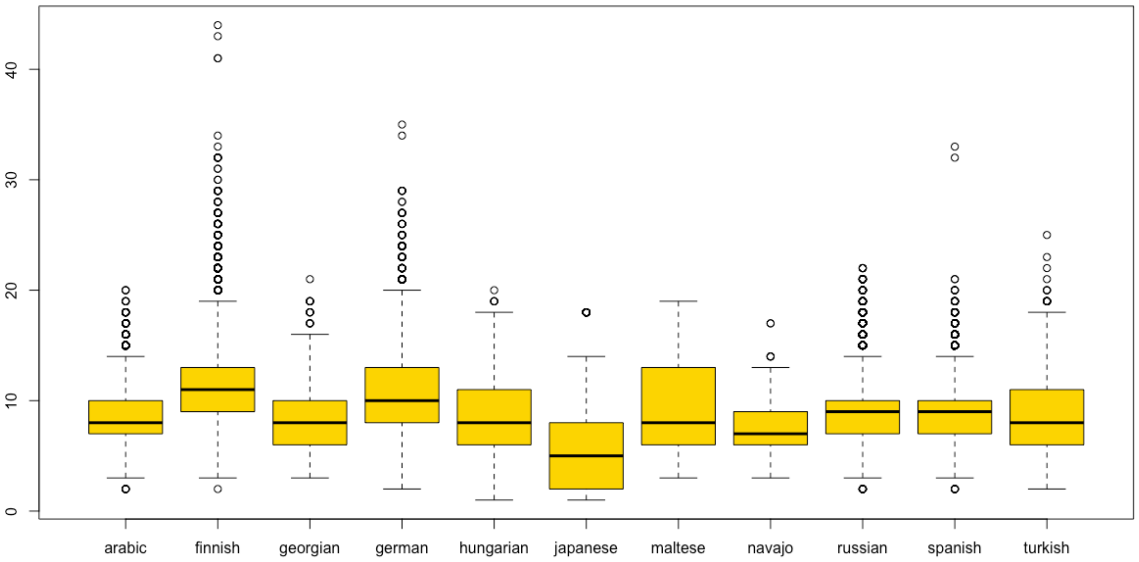
	Word length	Number of different words	Number of different transformations	Number of pairs per transformation	Levenshtein distance
Arabic	8.2±2.2	16,002	226	65.1±22.0	3.8±1.6
Finnish	11.0±3.6	37,857	95	247.8±191.4	3.4±1.6
Georgian	8.1±2.2	19,722	90	192.8±385.0	1.8±1.2
German	10.4±3.5	19,955	99	153.9±75.9	2.2±1.3
Hungarian	8.7±3.0	3,452	82	26.8±28.5	3.6±1.3
Japanese	5.0±3.2	1,573	20	791.3±80.6	6.2±1.6
Maltese	9.4±3.6	3,428	1507	0.6±0.9	7.0±2.3
Navajo	7.7±2.1	622	42	9.9±10.0	3.8±2.1
Russian	8.8±2.8	29,871	83	268.1±286.5	1.8±1.1
Spanish	8.9±2.3	28,230	83	278.9±192.2	2.2±1.3
Turkish	8.8±3.6	2,686	167	8.5±5.9	5.3±2.3

Table B.2: Statistics about the test sets compared to the training sets.

	Portion of new words	Number of new transformations	Number of characters in the training set	Number of characters in the test set	Number of new characters
Arabic	86,81%	6	43	43	0
Finnish	84,02%	0	32	30	1
Georgian	83,5%	0	34	34	0
German	77,22%	2	58	57	0
Hungarian	65,64%	0	33	33	0
Japanese	0%	0	632	632	0
Maltese	70,54%	105	30	30	1
Navajo	60,61%	0	30	30	0
Russian	78,53%	2	34	35	1
Spanish	87,03%	0	33	33	0
Turkish	58,90%	0	35	34	0

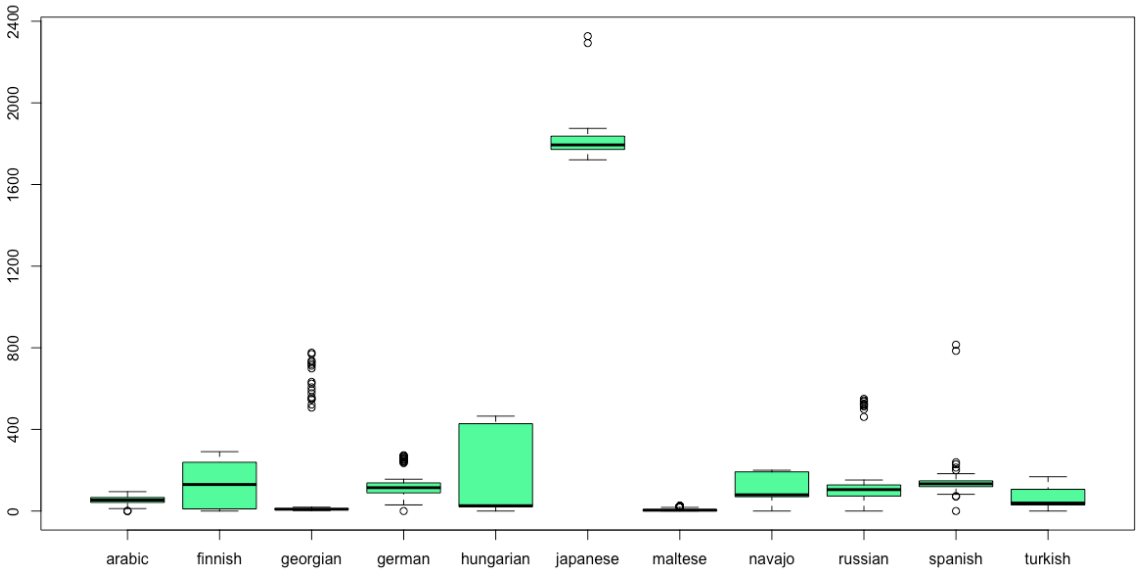


(a) Lengths of the words in the training sets.

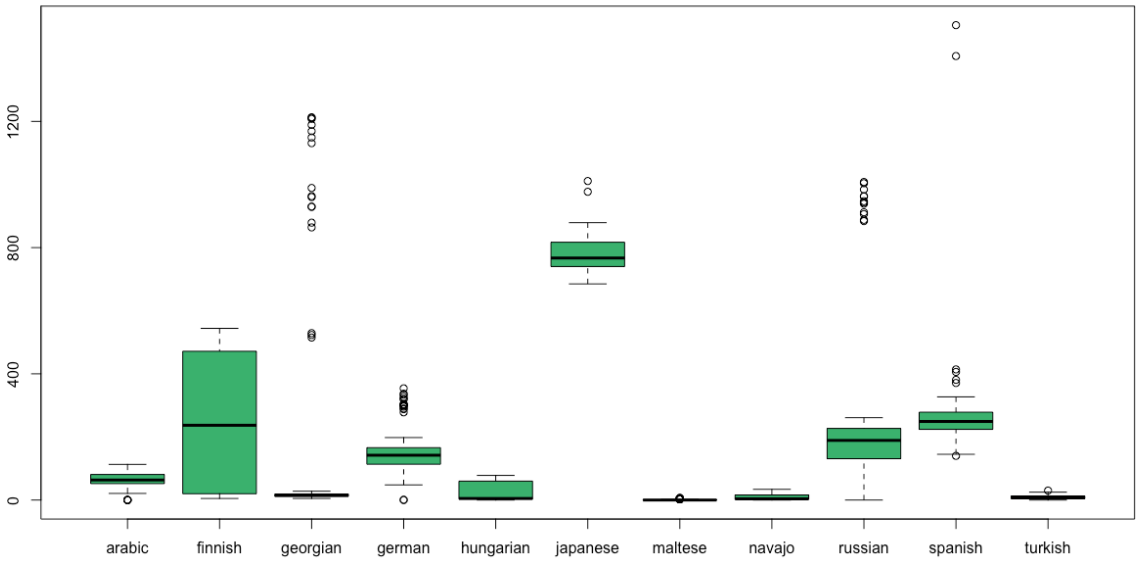


(b) Lengths of the words in the test sets.

Figure B.1: Lengths of the words.



(a) Number of word pairs per transformation in the training sets.



(b) Number of word pairs per transformation in the test sets.

Figure B.2: Number of word pairs per transformation.

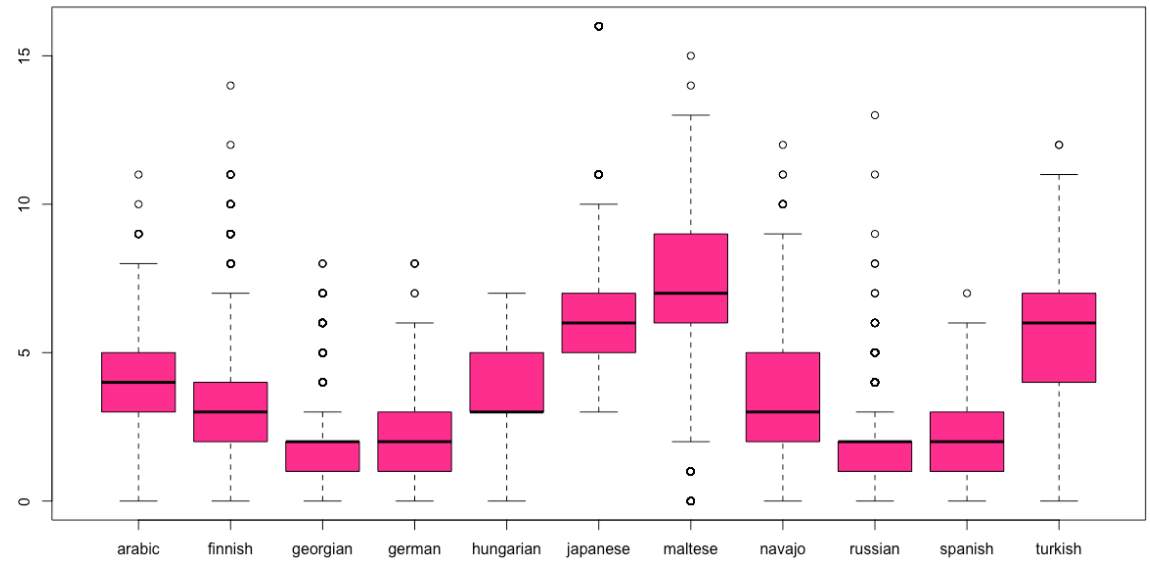


Figure B.3: Levenshtein distance between the words of the training set's pairs.
The values are similar for the test set

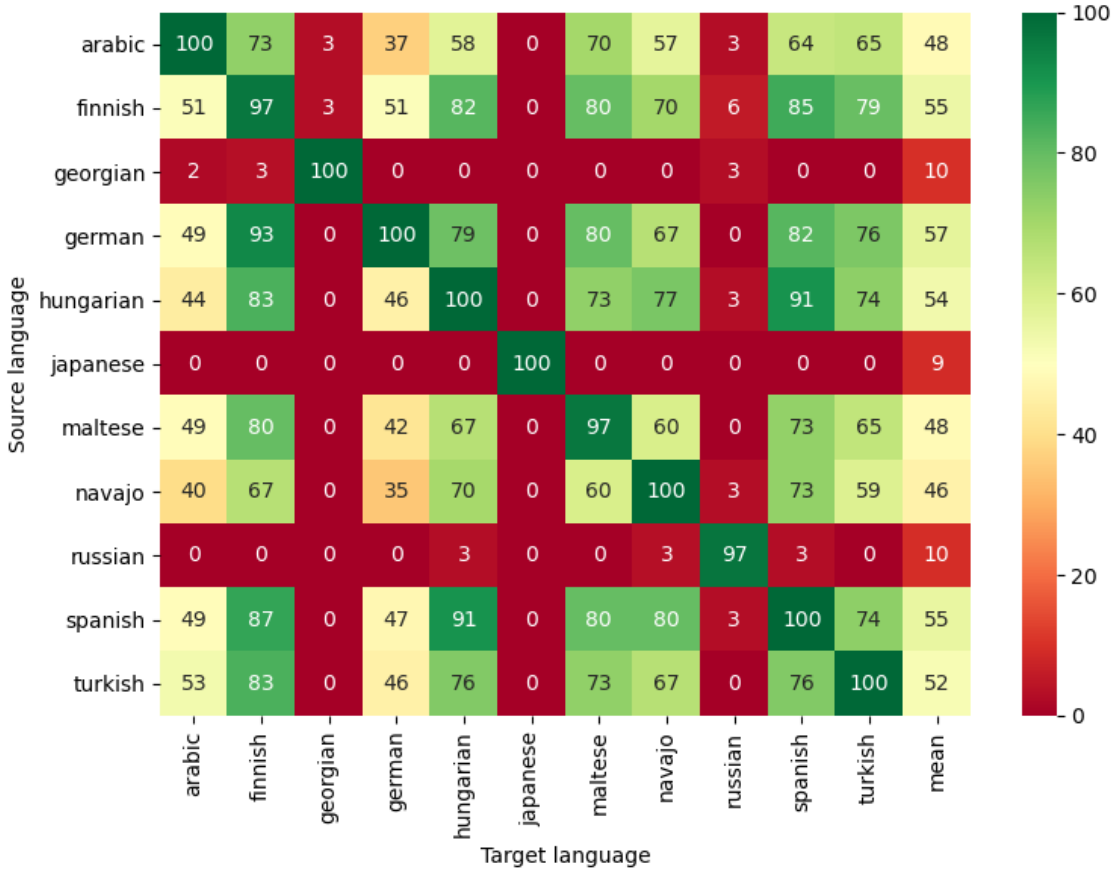


Figure B.4: Vocabulary coverage (in %) of the models on the test sets.

Appendix C

Examples for the classification and analogy solving tasks

In this appendix we propose examples in each language that we worked on in this project. Appendix C.1 is dedicated to the classification task while Appendix C.2 presents examples for solving analogies.

For the classification examples, we provide five examples per language:

- A valid one classified as valid by our model (true positive);
- A valid one classified as invalid by our model (false negative);
- An invalid one classified as invalid by our model (true negative);
- An invalid one classified as valid by our model (false positive);
- A random one (thus invalid) classified as invalid by our model.

For the solving task, we propose three examples per language:

- One where the right vector was the closest to the vector produced by our model;
- One where the right vector was found but was not the closest;
- One where the right vector was not found.

We display the words corresponding to the vectors close to the predicted one up to a 2%-extended distance. For the examples where the right vector was not found, it was not found with $k = 5\%$ either.

C.1 Classification task

	Expected	Result	Analogy	Form
Arabic	valid	valid	naffaqa:naffaqnā::dammama:dammamnāā	A:B::C:D
	valid	invalid	bayyāᶜūna:al-bayyāᶜu::nawarun:an-nawariyyu	B:A::D:C
	invalid	invalid	dammama:naffaqnā::naffaqa:dammamnā	C:B::A:D
	invalid	valid	al-ˀamtiᶜatu:al-matāᶜu::al-qīmatu:al-qiyamu	B:A::C:D
	invalid	invalid	jihādiyyayni:al-ḵanādiqu::tušarraḥa:muṣṭabiḥ	Random
Finnish	valid	valid	lenkkitossut:lenkkitossuilla::kananaivo:kananaivoilla	A:B::C:D
	valid	invalid	asukastiheys:asukastiheydeksi::aamutuima:aamutuimaksi	A:B::C:D
	invalid	invalid	kananaivo:lenkkitossuilla::lenkkitossut:kananaivoilla	C:B::A:D
	invalid	valid	raanin:raani::haartätäkki:haartätäkin	B:A::C:D
	invalid	invalid	juurimme:borderterrierin::pedannee:monstrumeilta	Random
Georgian	valid	valid	ინგლისელი:ინგლისელნო::პროლეტარი:პროლეტარნო	A:B::C:D
	valid	invalid	ინგლისელით:ინგლისელნო::ინდუსმა:ინდუსნო	A:B::C:D
	invalid	invalid	ინგლისელნო:ინგლისელი::პროლეტარი:პროლეტარნო	B:A::C:D
	invalid	valid	სათნო:სათნო::დანჯღრეული:დანჯღრეულმა	A:A::C:D
	invalid	invalid	მოლიბდენი:ტკივილი::საცობი:ტორფიანი	Random
German	valid	valid	extrovertiert:extrovertiertere::angelsächsisch:angelsächsischere	A:B::C:D
	valid	invalid	entgehen:entgingen::schwächen:schwächten	A:B::C:D
	invalid	invalid	extrovertiertere:extrovertiert::angelsächsisch:angelsächsischere	B:A::C:D
	invalid	valid	unkommunikative:unkommunikativ::abgestrahlt:abgestrahlte	B:A::C:D
	invalid	invalid	kultivierte:Wemfall::barbusigen:verwirklicht	Random
Hungarian	valid	valid	felejt:felejtenétek::elvét:elvétenétek	A:B::C:D
	valid	invalid	kert:kertre::cél:célra	A:B::C:D
	invalid	invalid	felejt:felejt::elvét:elvétenétek	A:A::C:D
	invalid	valid	felfog:felejtenétek::felejt:felfognátok	A:D::C:B
	invalid	invalid	nyissak:kezdeményezel::lokalizálhat:visszhangoztok	Random
Japanese	valid	valid	表現: 表現力/ひょうげんりょく:: 英語: 英語力/えいごりょく	A:B::C:D
	valid	invalid	盛り: 大盛り/おおもり:: 動脈: 大動脈/だいどうみゃく	A:B::C:D
	invalid	invalid	英語: 表現力/ひょうげんりょく:: 表現: 英語力/えいごりょく	A:D::C:B
	invalid	valid	持ち込ま: 持と/もと:: 持た: 持ち込も/もちこも	A:D::C:B
	invalid	invalid	筆者/ひっしゃ: 良い:: 会う: 馬	Random

Table C.1: First set of examples for the classification task.

	Expected	Result	Analogy	Form
Maltese	valid	valid	twassal:jitwassluhulu::tbelles:jitbellsuhulu	A:B::C:D
	valid	invalid	sempel:ssempel::ttertaq:tittertaq	A:B::C:D
	invalid	invalid	twassal:twassal::tbelles:jitbellsuhulu	A:A::C:D
	invalid	valid	lliżżem:liżżem::ballas:nballas	B:A::C:D
	invalid	invalid	tintilefhomlhiex:tnaqqithielna::berrinnihilhomx:jitħallaqhomlniex	Random
Navajo	valid	valid	yiiɉɉh:deiɉɉ ::yi éés:daaz eez	A:B::C:D
	valid	invalid	dajiiɉɉ :deiɉɉ ::yiyíiłdzid:dayíiłdzid	A:B::C:D
	invalid	invalid	dasiilghał:yiyiiłhał::ch íił á:ch ídaniil a	B:A::C:D
	invalid	valid	haidizóóh:daahshóóh::yishóóh:hadadohsóóh	A:D::C:B
	invalid	invalid	woozhizh :akétal ::hashohkeeh:ni siidzo	Random
Russian	valid	valid	доверчивость:доверчивостях::кровь:кровяхъ	A:B::C:D
	valid	invalid	выбивать:выбить::различать:различить	A:B::C:D
	invalid	invalid	доверчивостях:доверчивость::сертификат:сертификатах	B:A::C:D
	invalid	valid	пожалуйтесь:пожаловаться::изготовить:изготовьте	B:A::C:D
	invalid	invalid	натиску:придаточными::настроил:списывала	Random
Spanish	valid	valid	gobernar:gobernabais::empapelar:empapelabais	A:B::C:D
	valid	invalid	asolear:asoleando::desbandar:debandando	A:B::C:D
	invalid	invalid	gobernabais:gobernar::empapelar:empapelabais	B:A::C:D
	invalid	valid	zarpe:zarpar::coprotagonizar:coprotagonice	B:A::C:D
	invalid	invalid	zocatos:calibraré::domicilios:pizarrines	Random
Turkish	valid	valid	yumurtalık:yumurtalığınıza::kiriş:kirişinize	A:B::C:D
	valid	invalid	deney:deney::garaj:garaj	A:A::C:C
	invalid	invalid	kirişi:yumurtalığınıza::yumurtalıkları:kirişinize	C:B::A:D
	invalid	valid	yumurtalık:yumurtalık::kiriş:kirişinize	A:A::C:D
	invalid	invalid	dedikoducuların:emmezler::kıblesi:gençlikte	Random

Table C.2: Second set of examples for the classification task.

C.2 Solving analogies

	Analogical equation	Expected	Rank
Arabic	al-ʾabzanu:al-ʾabzanayni::az-zaᶜīmu::?	az-zaᶜīmayni	1
	$\xrightarrow[k=2\%]{Results}$ **az-zaᶜīmayni**, al-māᶜizayni, al-māᶜizatayni, al-maᶜzatayni, al-qirmiziyyayni, al-quzīmiyyayni, al-qamīṣayni, al-qirmiziyyatayni, al-quzīmiyyatayni		
	al-ʾabzanayni:al-ʾabzanu::az-zaᶜīmayni::?	az-zaᶜīmu	4
	$\xrightarrow[k=2\%]{Results}$ al-maᶜzatu, al-māᶜizatu, al-māᶜizu, **az-zaᶜīmu**, al-mawāᶜizu, al-qirmiziyyatu, al-maᶜlūmu		
	az-zaᶜīmayni:az-zaᶜīmu::al-ʾabzanayni::?	al-ʾabzanu	Not found
	$\xrightarrow[k=2\%]{Results}$ al-juzayʾu, al-ʾiblīsu, al-ʾaṯlāṯu, al-baḵšīšu, al-juzʾu		
Finnish	kaksiavioisuus:kaksiavioisuuksille::luotiliivi::?	luotiliiveille	1
	$\xrightarrow[k=2\%]{Results}$ **luotiliiveille**, luottaisitte, luokille, luotitte, lusikoitte		
	pojo:hemmo::pojoa::?	hemmoa	3
	$\xrightarrow[k=2\%]{Results}$ hemmoilta, hemmoista, **hemmoa**		
	päitset:päitset::perfektit::?	perfekti	Not found
	$\xrightarrow[k=2\%]{Results}$ perfektit, perhostelevat, perkussionistit, perhostelet, perennoisit		
Georgian	მაგნიტიზმი:მაგნიტიზმი::ტრაქტორი::?	ტრაქტორი	1
	$\xrightarrow[k=2\%]{Results}$ **ტრაქტორი**, ტრაქტატი, ტრაქტი		
	კალვინისტი:კალვინისტის::პატრიოტიზმი::?	პატრიოტიზმის	2
	$\xrightarrow[k=2\%]{Results}$ პატრიოტიზმს, **პატრიოტიზმის**, პატრიოტიზმით, პატრიარქის		
	უფულო:უფულო::პირუტყვულმა::?	პირუტყვული	Not found
	$\xrightarrow[k=2\%]{Results}$ პირუტყვულმა, პირუტყვულო, პირუტყვმა		

Table C.3: First set of examples for the solving task.

	Analogical equation	Expected	Rank
German	toxisch:populär::toxischere::?	populärere	1
	$\xrightarrow[k=2\%]{Results}$ **populärere**, populäre		
	ausgestorben:ausgestorbenes::mitverantwortlich::?	mitverantwortliches	2
	$\xrightarrow[k=2\%]{Results}$ mitverantwortlicher, **mitverantwortliches**, mitverantwortliche		
	applizierten:applizieren::versähen::?	versehen	Not found
	$\xrightarrow[k=2\%]{Results}$ versähen, verschlingen, verschieben		
Hungarian	bizonyít:taszít::bizonyítsak::?	taszítsak	1
	$\xrightarrow[k=2\%]{Results}$ **taszítsak**, taszítottuk, taszítok, taszítsatok		
	felfrissít:felfrissítsünk::döf::?	döfjünk	2
	$\xrightarrow[k=2\%]{Results}$ döfnék, **döfjünk**, döfnénk, döfjük, döftünk		
	adminisztrál:adminisztráltatok::kivisz::?	kivittetek	Not found
	$\xrightarrow[k=2\%]{Results}$ kiviszitek, kivisszük		
Japanese	遊べ/あそべ: 従え/したがえ:: 遊ぶ::?	従う	1
	$\xrightarrow[k=2\%]{Results}$ **従う**, 活発, 従え, 寒い, 政治, 役員, 上映, 懇談, 景況, 委員, 動脈		
	集める/あつめる: 集まる:: 折る/おる::?	折れる	4
	$\xrightarrow[k=2\%]{Results}$ 温まる, 探る, 閉まる, **折れる**, 決まる		
	縮まる/ちぢまる: 縮む:: 下ろす/くだろす::?	下りる	Not found
	$\xrightarrow[k=2\%]{Results}$ 過ぎる, 抑えよ, 探る, 倒れる, 破れる, 探れ, 表れる, 壊れる		
Maltese	tliegħeb:tlegħibniehielek::tliegħeb::?	tlegħibniehielek	1
	$\xrightarrow[k=2\%]{Results}$ **tlegħibniehielek**, tgħabbiniehielek, nitliegħbuhielek, tbekbkithilhiex		
	tlewnitilkomx:tlewwen::żżegħblitilkomx::?	żżegħber	2
	$\xrightarrow[k=2\%]{Results}$ taħbiża, **żżegħber**, żżambar		
	jiġtirruhulu:jitqaxilfuhulu::iġtar::?	tqaxlef	Not found
	$\xrightarrow[k=2\%]{Results}$ nqafel, tqaċċat, xaqqaf, tlaqqat		

Table C.4: Second set of examples for the solving task.

	Analogical equation	Expected	Rank
Navajo	ach$^{\text{ɔ}}$oozhlaa$^{\text{ɔ}}$:nich$^{\text{ɔ}}$oozhlaa$^{\text{ɔ}}$::ach$^{\text{ɔ}}$íí$^{\text{ɔ}}$::? $\xrightarrow[k=0.02]{Results}$ **nich$^{\text{ɔ}}$íí$^{\text{ɔ}}$**, nihich$^{\text{ɔ}}$íí$^{\text{ɔ}}$, yinichii$^{\text{ɔ}}$	nich$^{\text{ɔ}}$íí$^{\text{ɔ}}$	1
	nich$^{\text{ɔ}}$oozhlaa$^{\text{ɔ}}$:ach$^{\text{ɔ}}$oozhlaa$^{\text{ɔ}}$::nich$^{\text{ɔ}}$íí$^{\text{ɔ}}$::? $\xrightarrow[k=0.02]{Results}$ hach$^{\text{ɔ}}$íí$^{\text{ɔ}}$, **ach$^{\text{ɔ}}$íí$^{\text{ɔ}}$**, shich$^{\text{ɔ}}$íí$^{\text{ɔ}}$	ach$^{\text{ɔ}}$íí$^{\text{ɔ}}$	2
	ataa$^{\text{ɔ}}$:álátsíín::nitaa$^{\text{ɔ}}$::? $\xrightarrow[k=0.02]{Results}$ nihílátsíín, nitł$^{\text{ɔ}}$in, nanimaas	nílátsíín	Not found
Russian	буфет:буфетах::архимандрит::? $\xrightarrow[k=0.02]{Results}$ **архимандритах**	архимандритах	1
	архимандрит:архимандритах::буфет::? $\xrightarrow[k=0.02]{Results}$ буфетов, **буфетах**, буфет	буфетах	2
	обновление:пароход::обновлений::? $\xrightarrow[k=0.02]{Results}$ пароход	пароходов	Not found
Spanish	repavimentar:repavimentará::encantar::? $\xrightarrow[k=0.02]{Results}$ **encantará**, encestara, encriptará, encintara	encantará	1
	bibliografías:bibliografía::turones::? $\xrightarrow[k=0.02]{Results}$ turbaren, turnaban, turnan, **turón**, turban	turón	4
	decodificásemos:decodificar::empalmásemos::? $\xrightarrow[k=0.02]{Results}$ empapelar, empalar, empollar, ampollar, empolvar, emplear, impeler, emplazar, espolear, empalagar	empalmarse	Not found
Turkish	kasık:kasıklarında::kasık::? $\xrightarrow[k=0.02]{Results}$ **kasıklarında**, kaslarında	kasıklarında	1
	mumya:mumyalarımızdan::film::? $\xrightarrow[k=0.02]{Results}$ filizleri, **filmlerimizden**, filizlerinde, filizlerinden, filmlerimden, filmlerinde, filmlerinden, filozoflarımızı, filizlerinizden, fillerdiniz, fitillerimizin, filmlerini, fizikçilerimizin, fillerinde, fillerini	filmlerimizden	2
	filmlerimizden:film::mumyalarımızdan::? $\xrightarrow[k=0.02]{Results}$ müellif, müellifi	mumya	Not found

Table C.5: Third set of examples for the solving task.

Appendix D

Analogy solving k-extended results

In this appendix, we present some statistics about the k-extended evaluation of the solving task.

Table D.1 contains the number of analogies for which we found more than one vector in the k-extended range. We also recall the total number of analogies used for the evaluation as a comparison.

Figure D.1 shows the rank of the right vector among the found ones in the 5%-extended range through boxplots. We chose to keep our y-axis in $[0, 150]$ for visibility reasons, there are thus outliers which do not appear on the chart.

Table D.2 present statistics about the sets and ranks of found vectors for the solving task with Cosine similarity. It contains the MRR, the mean of the ranks when the rank is greater than one ($\pm$ standard deviation), the portion of sets of found vectors of length greater than one and the mean size of the sets ($\pm$ standard deviation).

Table D.3 present the same statistics but with Euclidean distance.

	Cosine similarity				Euclidean distance				Total number
	k = 0	k = 0.01	k = 0.02	k = 0.05	k = 0	k = 0.01	k = 0.02	k = 0.05	of analogies
Arabic	1	269,116	360,932	399,882	0	48,553	89,706	182,878	400,000
Finnish	2	237,609	350,193	399,904	0	25,320	47,542	102,442	400,000
Georgian	0	79,355	180,971	385,976	0	4,415	8,610	20,666	400,000
German	0	86,346	185,771	385,026	0	8,234	15,762	36,542	400,000
Hungarian	0	185,849	253,444	374,526	0	25,706	49,064	105,364	400,000
Japanese	3	57,525	62,522	63,384	0	13,288	23,568	43,085	63,384
Maltese	0	13,374	19,268	25,188	0	1,708	3,164	6,432	29,656
Navajo	0	16,508	26,164	36,808	0	3,682	6,982	14,572	38,744
Russian	3	92,499	171,110	341,368	0	16,788	32,640	73,566	400,000
Spanish	8	130,858	229,238	377,306	6	11,836	22,656	51,100	400,000
Turkish	0	32,370	54,742	84,412	0	5,768	10,778	23,752	90,880

Table D.1: Number of analogies for which we found more than one vector in the k-extended range.

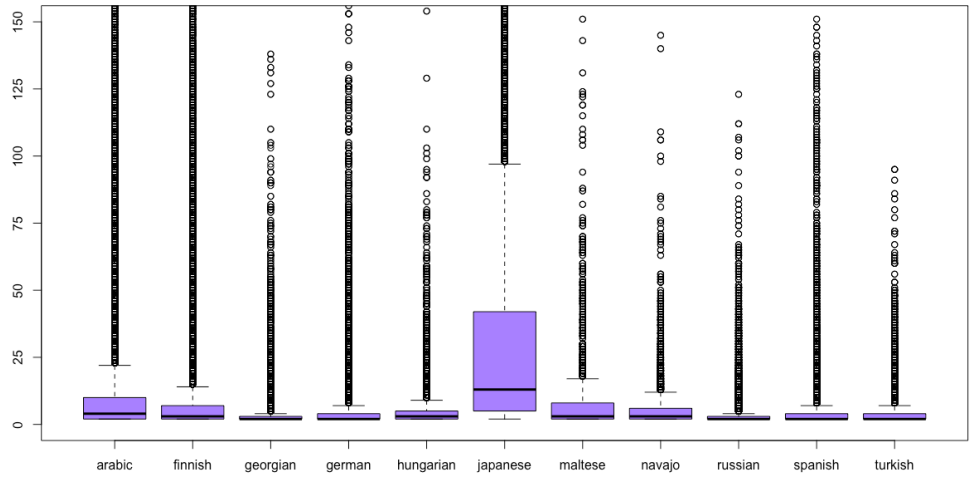


Figure D.1: Rank of the right vector among the found ones in the 5%-extended range.
For visibility reasons, we did not display all the outliers (they reach 3601 for Maltese).

	MRR			**MR if rank > 1**		
	k = 0.01	k = 0.02	k = 0.05	k = 0.01	k = 0.02	k = 0.05
Arabic	0.60	0.63	0.64	2.9 ± 1.6	4.4 ± 4.3	12.2 ± 28.4
Finnish	0.80	0.81	0.81	3.1 ± 2.0	4.6 ± 5.2	9.8 ± 26.2
Georgian	0.95	0.95	0.96	2.1 ± 0.5	2.4 ± 1.2	4.2 ± 8.2
German	0.90	0.91	0.92	2.3 ± 0.7	2.7 ± 1.8	5.3 ± 12.1
Hungarian	0.76	0.77	0.78	2.9 ± 1.4	3.4 ± 2.1	4.1 ± 4.0
Japanese	0.28	0.30	0.30	32.3 ± 53.8	38.4 ± 77.3	39.7 ± 72.9
Maltese	0.79	0.79	0.80	4.6 ± 6.2	8.0 ± 21.4	17.8 ± 124.5
Navajo	0.51	0.54	0.58	2.4 ± 0.9	3.0 ± 1.7	5.3 ± 7.4
Russian	0.73	0.76	0.80	2.2 ± 0.5	2.4 ± 1.0	3.3 ± 4.1
Spanish	0.92	0.92	0.92	2.5 ± 1.1	3.0 ± 2.7	5.3 ± 14.2
Turkish	0.75	0.77	0.79	2.3 ± 0.8	2.7 ± 1.4	4.4 ± 5.7
	Portion of sets of length > 1			**Mean size of the sets**		
	k = 0.01	k = 0.02	k = 0.05	k = 0.01	k = 0.02	k = 0.05
Arabic	67.28%	90.23%	99.97%	3.9 ± 2.5	8.9 ± 8.4	100.4 ± 111.2
Finnish	59.40%	87.55%	99.98%	4.4 ± 3.4	11.1 ± 11.5	153.7 ± 137.9
Georgian	19.84%	45.24%	96.49%	2.3 ± 0.7	2.8 ± 1.7	12.1 ± 13.4
German	21.59%	46.44%	96.26%	2.5 ± 1.1	3.3 ± 2.8	16.0 ± 25.6
Hungarian	46.46%	63.36%	93.63%	3.7 ± 2.0	5.4 ± 3.6	14.0 ± 20.0
Japanese	90.76%	98.64%	100.00%	136.1 ± 140.6	329.1 ± 332.7	461.8 ± 362.3
Maltese	45.10%	64.97%	84.93%	8.0 ± 11.9	52.2 ± 91.9	1376.4 ± 1785.4
Navajo	42.61%	67.53%	95.00%	2.7 ± 1.2	3.9 ± 2.8	13.7 ± 16.8
Russian	23.12%	42.78%	85.34%	2.3 ± 0.8	2.8 ± 1.6	7.2 ± 9.7
Spanish	32.71%	57.31%	94.33%	3.0 ± 1.7	5.0 ± 5.0	34.5 ± 48.7
Turkish	35.62%	60.24%	92.88%	2.6 ± 1.1	3.7 ± 2.7	18.2 ± 25.4

Table D.2: Results of the k-extended evaluation for analogical equation solving with Cosine similarity

	MRR			**MR if rank > 1**		
	k = 0.01	k = 0.02	k = 0.05	k = 0.01	k = 0.02	k = 0.05
Arabic	0.53	0.54	0.58	2.1±0.3	2.2±0.5	2.6±1.1
Finnish	0.73	0.74	0.76	2.1±0.3	2.1±0.4	2.4±0.9
Georgian	0.94	0.94	0.94	2.0±0.1	2.0±0.2	2.1±0.5
German	0.88	0.88	0.89	2.0±0.2	2.1±0.3	2.2±0.7
Hungarian	0.69	0.70	0.72	2.1±0.3	2.1±0.4	2.3±0.7
Japanese	0.18	0.19	0.21	2.2±0.5	2.4±0.7	3.2±1.7
Maltese	0.77	0.78	0.79	2.1±0.4	2.2±0.6	2.6±1.2
Navajo	0.49	0.50	0.52	2.1±0.3	2.2±0.4	2.5±1.0
Russian	0.70	0.70	0.72	2.0±0.2	2.1±0.3	2.2±0.7
Spanish	0.88	0.89	0.90	2.0±0.2	2.1±0.3	2.2±0.6
Turkish	0.69	0.69	0.72	2.1±0.2	2.1±0.4	2.3±0.8
	Portion of sets of length > 1			**Mean size of the sets**		
	k = 0.01	k = 0.02	k = 0.05	k = 0.01	k = 0.02	k = 0.05
Arabic	12.14%	22.43%	45.72%	2.2±0.4	2.3±0.7	3.1±1.7
Finnish	6.33%	11.89%	25.61%	2.1±0.4	2.2±0.6	2.7±1.3
Georgian	1.10%	2.15%	5.17%	2.0±0.2	2.1±0.3	2.2±0.8
German	2.06%	3.94%	9.14%	2.1±0.3	2.1±0.5	2.4±1.1
Hungarian	6.43%	12.27%	26.34%	2.1±0.3	2.2±0.5	2.6±1.0
Japanese	20.96%	37.18%	67.97%	2.3±0.6	2.6±0.9	4.0±2.4
Maltese	5.76%	10.67%	21.69%	2.2±0.5	2.4±0.8	3.2±1.9
Navajo	9.50%	18.02%	37.61%	2.1±0.3	2.3±0.6	2.8±1.4
Russian	4.20%	8.16%	18.39%	2.1±0.3	2.2±0.5	2.5±1.1
Spanish	2.96%	5.66%	12.78%	2.1±0.3	2.2±0.5	2.4±1.1
Turkish	6.35%	11.86%	26.14%	2.1±0.3	2.2±0.6	2.6±1.2

Table D.3: Results of the k-extended evaluation for analogical equation solving with Euclidean distance.