Neural Computing and Applications
https://doi.org/10.1007/s00521-022-07164-x

ORIGINAL ARTICLE

Cross-lingual alignments of ELMo contextual embeddings

Matej Ulčar[1] · Marko Robnik-Šikonja[1]

Received: 26 July 2021 / Accepted: 2 March 2022

Abstract
Building machine learning prediction models for a specific natural language processing (NLP) task requires sufficient training data, which can be difficult to obtain for less-resourced languages. Cross-lingual embeddings map word embeddings from a less-resourced language to a resource-rich language so that a prediction model trained on data from the resource-rich language can also be used in the less-resourced language. To produce cross-lingual mappings of recent contextual embeddings, anchor points between the embedding spaces have to be words in the same context. We address this issue with a novel method for creating cross-lingual contextual alignment datasets. Based on that, we propose several cross-lingual mapping methods for ELMo embeddings. The proposed linear mapping methods use existing Vecmap and MUSE alignments on contextual ELMo embeddings. Novel nonlinear ELMoGAN mapping methods are based on generative adversarial networks (GANs) and do not assume isomorphic embedding spaces. We evaluate the proposed mapping methods on nine languages, using four downstream tasks: named entity recognition (NER), dependency parsing (DP), terminology alignment, and sentiment analysis. The ELMoGAN methods perform very well on the NER and terminology alignment tasks, with a lower cross-lingual loss for NER compared to the direct training on some languages. In DP and sentiment analysis, linear contextual alignment variants are more successful.

Keywords Contextual embeddings · Cross-lingual models · ELMo · GAN · Non-isomorphic contextual alignment · Non-linear vector alignment

1 Introduction

Word embeddings are representations of words in a numerical form, as vectors of typically several hundred dimensions. The vectors are used as inputs to machine learning models; these are generally deep neural networks for complex language processing tasks. The embedding vectors are obtained from specialized neural network-based embedding algorithms. The quality of embeddings depends on the amount of semantic information expressed in the embedded space through distances and directions. For that reason, static pre-trained word embeddings, such as word2vec [1] or fastText [2], have in large part been recently replaced by contextual embeddings, such as ELMo [3] and BERT [4].

Contextual embeddings generate a different word vector for the same word for every context it appears in. BERT models and their derivatives are mostly used as a closed system, where the entire model is fine-tuned on a downstream task. On the other hand, ELMo models typically generate different word vectors for each word occurrence, and these vectors are used in training natural language processing (NLP) models. A neural network producing ELMo embeddings contains three layers of neurons. Embeddings are typically a concatenation of network weights in all three layers. BERT models consist of 12 or 24 layers, and vector extraction typically uses a combination of only the last four layers. Due to the omission of most network layers, explicit BERT vectors may lack a lot of information. For that reason, the explicit BERT vectors are rarely used and are often less successful than ELMo vectors, see, e.g., [5]. A smaller size of ELMo models compared to BERT, may also offer better explainability of the end-task models.

✉ Matej Ulčar
matej.ulcar@fri.uni-lj.si

Marko Robnik-Šikonja
marko.robnik@fri.uni-lj.si

1 Faculty of Computer and Information Science, University of Ljubljana, Večna pot 113, Ljubljana, Slovenia

Modern word embedding spaces exhibit similar structures across languages, even when considering distant language pairs like English and Vietnamese [1]. This means that embeddings independently produced from monolingual text resources can be aligned, resulting in a common cross-lingual representation, called cross-lingual embeddings, which allows for fast and effective integration of information in different languages. For less-resourced languages, training NLP models can be difficult because of a lack of data for a specific task. The aim of cross-lingual alignment is to use an already existing model trained on a resource-rich language and map the word embeddings from a less-resourced language vector space to the resource-rich language vector space. In that way, the input in less-resourced language is mapped to resource-rich language and can be classified with existing models in that language. This is possible as the words with the same meaning in both languages have very similar vectors after the cross-lingual alignment.

Cross-lingual approaches can be sorted into several groups. The first group of methods uses monolingual embeddings with (an optional) help from bilingual dictionaries to align the embeddings. These methods are typically used for static embeddings, such as word2vec and fastText. The second group of approaches uses bilingually aligned (comparable or even parallel) corpora for joint construction of embeddings in all involved languages. An example of this approach is a joint space of 93 languages produced by the LASER library [6]. The third type of approaches is based on large pretrained multilingual masked language models such as BERT [4], which were simultaneously trained on many languages and work reasonably well in cross-lingual transfer [7] without the need for explicit cross-lingual alignment. In this work, we present an extension of the first group of approaches to contextual embeddings. We focus on improvements of cross-lingual mappings for ELMo contextual embeddings (BERT, being multilingual, does not need them). Currently, the most successful alignment methods assume that the embedding spaces in different languages are isomorphic [8, 9], which is generally not the case. Researchers have observed that the monolingual embedding spaces of two different languages are not completely isomorphic, which is especially true for distant languages [10, 11]. As a result, many of these methods are unstable or unsuccessful when confronted with distant language pairs.

We propose novel methods for linear and nonlinear alignment of contextual embeddings, such as ELMo. For that purpose, we first construct novel contextual mapping datasets based on parallel corpora and dictionaries. In the novel ELMoGAN approach, we use generative adversarial networks (GANs) [12], that produce nonlinear mappings between the embedding spaces. The main contributions of this work are as follows.

1. A novel approach to create datasets needed in the cross-lingual alignment of contextual embeddings.
2. Novel linear and nonlinear cross-lingual alignment methods for ELMo embeddings.
3. Evaluation of produced cross-lingual embeddings using nine languages and four downstream tasks: named entity recognition (NER), dependency parsing (DP), terminology alignment, and sentiment analysis.

The results show a successful cross-lingual transfer of tested approaches. The best alignment method depends on the task. We publish the code to create the needed datasets and the code to produce nonlinear cross-lingual alignment under a permissive license[1].

The paper is split into five further sections. In Sect. 2, we present the background on cross-lingual alignment and ELMo and cover related work on cross-lingual embeddings. The construction of special datasets used for training the alignments of contextual embeddings is presented in Sect. 3. In Sect. 4, we describe the proposed nonlinear ELMoGAN alignment method and linear methods combining the new dataset and existing mapping methods. In Sect. 5, we evaluate the proposed alignment methods on four downstream tasks. We summarize our work in Sect. 6 and discuss opportunities for further work.

2 Background and related work

Word embeddings represent each word in a language as a vector in a high dimensional vector space so that the relations between words in a language are reflected in their corresponding embeddings. Cross-lingual embeddings attempt to map words represented as vectors from one vector space to the other so that vectors representing words with the same meaning in both languages are as close as possible. Søgaard et al. [13] present a detailed overview and taxonomy of cross-lingual methods.

In Sect. 2.1, we describe how two monolingual embedding spaces can be aligned with the optional help from a bilingual dictionary. This work's main focus is extending existing approaches that work with non-contextual embeddings to contextual ELMo embeddings. For this reason, we present the background on ELMo contextual embeddings in Sect. 2.2. The related work on non-contextual mappings is given in Sect. 2.3, and on contextual mappings in Sect. 2.4.

[1] https://github.com/MatejUlcar/elmogan.

2.1 Alignment of monolingual embeddings

Cross-lingual alignment methods take precomputed word embeddings for each language and align them with the optional use of bilingual dictionaries. Two types of monolingual embedding alignment methods exist. The methods of the first type map vectors representing words in one of the languages into the vector space of the other language. The methods of the second type map embeddings from both languages into a common vector space. The goal of both types of alignments is the same: the embeddings for words with the same meaning must be as close as possible in the final vector space. A comprehensive summary of existing approaches can be found in works by Artetxe et al. [14]. The open source implementation of the method described by Artetxe et al. [8, 14], named *Vecmap*[2], is able to align monolingual embeddings using supervised, semi-supervised or unsupervised approach.

The supervised approach requires a large bilingual dictionary, which is used to match embeddings of the words with the same meaning. The embeddings are aligned using the Moore-Penrose pseudo-inverse, which minimizes the sum of squared Euclidean distances. The algorithm always converges but can be caught in a local maximum when the initial solution is poor. To overcome this, several methods (stochastic dictionary introduction, frequency-based vocabulary cutoff, etc.) are used that help the algorithm to climb out of local maxima. A more detailed description of the algorithm is given in [8].

The semi-supervised approach uses a small initial seeding dictionary, while the unsupervised approach is run without any bilingual information. The latter uses similarity matrices of both embeddings to build an initial dictionary. This initial dictionary is usually of poor but sufficient quality for later processing. After the initial dictionary (either by seeding dictionary or using similarity matrices) is built, the iterative algorithm is applied. The algorithm first computes an optimal mapping using the pseudo-inverse approach for the given initial dictionary. Then optimal dictionary for the given embeddings is computed, and the procedure is repeated with the new dictionary.

Besides *Vecmap*, another well-known library for cross-lingual embeddings alignment is called MUSE[3]. This library can find a cross-lingual map with the use of a bilingual dictionary (supervised) or without one (unsupervised approach). The unsupervised approach works by using adversarial training to find the starting linear mapping. A synthetic dictionary is extracted from this mapping, which is used to fine-tune the starting mapping using the Procrustes approach, described in detail by Conneau et al. [9].

[2] https://github.com/artetxem/vecmap.

[3] https://github.com/facebookresearch/MUSE.

2.2 ELMo contextual embeddings

ELMo (Embeddings from Language Models) embedding [3] is an example of a state-of-the-art pre-trained contextual embedding model. It is a neural network model, composed of three layers. The first layer is a convolutional neural network (CNN) layer, which operates on a character level. It is context-independent, so each word always gets the same embedding from this layer, regardless of its context. It is followed by two biLM (bidirectional language model) layers. A biLM layer consists of two concatenated long short-term memory layers (LSTMs) [15]. In the first LSTM, the network predicts the following word, based on the given past words, where the embeddings from the CNN layer represent each word. In the second LSTM, the network predicts the preceding word based on the given following words. This layer is equivalent to the first LSTM, just reading the text in reverse.

The actual embeddings are constructed from the internal states of the bidirectional LSTM neural network. Two higher-level LSTM layers capture context-dependent aspects, while the first CNN layer captures aspects of syntax [3]. To train the ELMo network, one puts one sentence at a time on the input. The representation of each word depends on the whole sentence, i.e., it reflects the contextual features of the input text and thereby polysemy of words. For an explicit word representation, one can use only the top layer. Still, more frequently, one combines all layers into a vector. The representation of a word or a token t_k at position k is composed of

$$R_k = \{x_k^{LM}, \overrightarrow{h}_{k,j}^{LM}, \overleftarrow{h}_{k,j}^{LM} \mid j = 1, \ldots, L\} \quad (1)$$

where L is the number of layers (ELMo uses $L = 2$), index j refers to the level of bidirectional LSTM network, x is the initial token representation (either word or character embedding), and h^{LM} denotes hidden layers of a forward or a backward language model. In NLP tasks, a weighted average of layers is usually used, where the weights are learned during the training of the model for the specific task. Alternatively, the entire ELMo model can be fine-tuned on a specific end-task.

At the time of its introduction, ELMo has been shown to outperform previous pre-trained word embeddings like word2vec and GloVe on many NLP tasks, e.g., question answering, named entity extraction, sentiment analysis, textual entailment, semantic role labelling, and coreference resolution [3]. Later, BERT models turned out to be even more successful on these tasks. However, concerning the quality of extracted vectors, ELMo can be advantageous as its information is condensed in only three layers. In comparison, the information in multilingual BERT is scattered over 12 layers [16, 17] and a reasonably sized embedding drops most of them. Peters et al. [18] found that the word vector extraction is advantageous for ELMo compared to fine-tuning, while the opposite is true for BERT. Furthermore, on some tasks, the ELMo embeddings outperform BERT [5, 19].

2.3 Related work on non-contextual mappings

Cross-lingual alignment methods align precomputed monolingual word embeddings from two or more languages. The word vectors from all the languages are mapped into a common vector space. This can be the same vector space as one of the original monolingual embeddings or a completely independent vector space. These methods aim to represent the words with the same meaning in different languages with as similar vectors as possible. Concerning the data used, the alignment methods can be split into supervised and unsupervised methods. Supervised methods determine the alignment of the embeddings with the use of bilingual dictionaries. Unsupervised methods do not use any bilingual data. Conneau et al. [9] trained the unsupervised alignment using adversarial training. Artetxe et al. [8] first constructed a low-quality seed dictionary using the assumption that the two vector spaces are isometric and then iteratively updated the mapping and dictionary until convergence.

Artetxe et al. [14] comprehensively summarize existing linear methods, showing that the state-of-the-art linear alignment methods can be summarized as an orthogonal mapping. The difference between various methods is solely due to different approaches to vector manipulation (such as normalization, whitening, etc.) before the mapping extraction.

Nakashole and Flauger [20] show that, in a small neighbourhood, linear mapping methods work well; however, the linearity assumption does not hold in general, especially for distant languages [10]. A few nonlinear alignment methods have been proposed. Lu et al. [21] trained nonlinear mapping using Deep Canonical Correlation Analysis (DCCA) [22], which is an expanded version of a linear Canonical Correlation Analysis (CCA) method, using deep neural networks. They showed that DCCA performs better than linear CCA. Recently, Zhao and Gilman [23] proposed a nonlinear mapping method, using kernel CCA (KCCA). KCCA projects the vectors into a higher dimensional space and then performs CCA in the new vector space. Zhao and Gilman [23] report that DCCA has to fine-tune many hyper-parameters and show that KCCA outperforms both DCCA and CCA, especially when data is scarce. In contrast, Lu et al. [21] observe that DCCA scales better with data size than KCCA.

Conneau et al. [9] used the adversarial training based on GANs to train a linear mapping between vector spaces. Yang et al. [24] have used full GAN models for neural machine translation. Fu et al. [25] trained a bidirectional GAN for cross-lingual alignment of sentence embeddings, improving the results over linear and nonlinear state-of-the-art methods on the sentence alignment task.

In contrast to the above works which deal with non-contextual embeddings, our work is focused on contextual ELMo embeddings. While we use some of the same techniques (GANs for nonlinear mappings and Procrustes approach for linear mappings), the task we tackle is different and much more difficult as we have to find a mapping for each context a word appears (and not just each word). The work on contextual mappings, closer to our approach, are outlined below.

2.4 Related work on contextual mappings

All the above work only concerned static embeddings, not dynamic, contextual embeddings. Schuster et al. [26] produced cross-lingual alignments of contextual ELMo embeddings. While each occurrence of a word in contextual embeddings is represented by a different vector, Schuster et al. [26] hypothesized that these vectors form clusters. Based on this assumption, they assigned each word a single static vector by calculating the average vector of all word occurrences in a large corpus. They used a linear MUSE method to calculate the alignments of the averaged vectors. This approach's problem is the assumption of isomorphic spaces and loss of information if this assumption is not true in the local context.

Aldarmaki and Diab [27] used parallel corpora to produce the embedding vectors. They aligned the corpora on the word level, using Fast Align [28], calculated the ELMo embeddings on the aligned corpora, and extracted a dictionary from the word-level alignments. Their approach showed good results in a sentence translation retrieval task. They measured the accuracy of retrieving the correct translation from the target side of a test parallel corpus using nearest neighbour search and cosine similarity. They applied their approach to three languages (English, German, Spanish). This approach is similar to the linear mappings applied to ELMo, which we describe in Sect. 4.3. The difference is that we use much larger dictionaries and test on many more language pairs. Our nonlinear approach, presented in Sects. 4.1 and 4.2, has no counterpart in the existing works.

3 Datasets for alignment of contextual embeddings

This section explains the training datasets produced for cross-lingual alignment of contextual ELMo embeddings. These datasets are essential for both linear and nonlinear mappings presented in Sect. 4. Besides the datasets, we also present the language resources used in their creation.

Supervised cross-lingual vector alignment approaches assume the existence of a bilingual dictionary, where each word from the dictionary has its own embedding vector. For static, non-contextual embeddings this is straightforward as one can take any of human- or machine-created dictionaries. In contextual embeddings, word vectors depend on the context words appears in. For every context, a word gets a different vector. Schuster et al. [26] approached this by averaging all the vectors of a given word, as described in Sect. 2. This approach loses information, as words have multiple meanings. For example, the word "bark" can refer to the sound a dog makes, a sailing boat, or the outer part of a tree trunk. Furthermore, two meanings may be represented with one word in one language but with two different words in another language.

We solve these issues by separately aligning each occurrence of a word. We start with a parallel corpus, aligned on a paragraph level to have matching contexts in two languages. Let i indicate the index of a context from a parallel corpus P. Let A and B represent the first and the second language in a language pair. Then P_i^A is the i-th paragraph/context from corpus P in language A. Given a bilingual dictionary D, let j indicate the index of a word pair in the dictionary so that the dictionary is composed of pairs (D_j^A, D_j^B), $\forall j \in \{1, \ldots, |D|\}$.

We construct our dataset by parsing the parallel corpus. For each word $a \in P_i^A$, we check whether its lemma appears in D^A. If it does, given its dictionary index j, we check whether D_j^B is a lemma of any word from P_i^B. If it is, we add the tuple

$$(iD_j^A, iD_j^B, e(D_j^A, P_i^A), e(D_j^B, P_i^B))$$

to our dataset, where $e(D_j^A, P_i^A)$ and $e(D_j^B, P_i^B))$ are (ELMo) embeddings of the two dictionary words D_j^A and D_j^B, computed in the context P_i for each of the languages, respectively. We considered at most 20 different contexts of each lemma to not overwhelm the dataset with frequent words (such as stop words). For the lemmatization of the corpora, we used the Stanza tool [29] in all analyzed languages. Note that we only used lemmatized corpora for dictionary look-up; for generating the embedding, we used the non-lemmatized corpora.

As we explained in Sect. 2.2, ELMo models are deep neural networks with three hidden layers. The first layer is a non-contextual CNN, followed by two contextual biLSTM layers. The final embedding vectors are constructed from vectors of all three layers. The first vector is contextually independent, while the second and third layers are contextually dependent. In the proposed cross-lingual alignment approaches for ELMo, we align vectors from each of the three layers separately. Thus, we need a separate dataset for each layer, i.e., iD_j^A, iD_j^B components of the learning tuples are the same in the datasets but $e(D_j^A, P_i^A), e(D_j^B, P_i^B)$ components are computed separately for each contextual LSTM layer. We created two such contextual datasets for each language pair, one for each of the contextual ELMo layers. For the non-contextual CNN layer, we produce embeddings for every word pair in the bilingual dictionary. As the non-contextual ELMo vectors are the same for all word contexts, the size of this dataset is identical to the bilingual dictionary size.

We split the created datasets into a training and evaluation part. We separately split data for each language pair and each ELMo layer. The training part has 98.5% of word vector pairs, and the evaluation part has 1.5% of word vector pairs.

In our work, we considered eleven language pairs from nine different languages. The language pairs along with the sizes of bilingual dictionaries, parallel corpora, and the final training dataset are presented in Table 1. For English, we used the original English 5.5B ELMo model[4]. For Russian, we used the ELMo model trained by DeepPavlov[5] on the Russian WMT News. For other seven languages, we used ELMo models trained by Ulčar and Robnik-Šikonja [30][6].

As sources of parallel texts, we used OpenSubtitles parallel corpora[7] [31] from the Opus web page[8] for each pair of languages. The dictionaries we used are bilingual dictionaries extracted from Wiktionary, using the wikt2dict[9] tool [32]. The tool allows for direct dictionary extraction, as well as triangulation via a third language. In the triangulation case, given three languages, A, B and C, one constructs a bilingual dictionary for languages A and B so that for every word $a \in A$, one finds its translation $c \in C$ from $A - C$ dictionary. Next, the translation of the word c in language B is found in the $C - B$ dictionary and labeled b. The dictionary created using triangulation consists of pairs $a - b$.

[4] https://allennlp.org/elmo.

[5] https://github.com/deepmipt/DeepPavlov.

[6] http://hdl.handle.net/11356/1277.

[7] https://www.opensubtitles.org/.

[8] http://opus.nlpl.eu.

[9] https://github.com/juditacs/wikt2dict.

Table 1 The sizes of dictionaries and parallel corpora used in the creation of ELMo contextual mapping datasets, as well as the size of the resulting datasets

Language pair	Type	Dictionary	Parallel corpus	ELMo dataset
English-Estonian	Direct	11,022	12,486,898	77,800
English-Finnish	Direct	89,307	27,281,566	283,000
English-Croatian	Direct	3448	35,131,729	44,800
English-Lithuanian	Direct	13,960	1,415,961	62,800
English-Latvian	Direct	10,224	519,553	43,800
English-Russian	Direct	103,850	25,910,105	363,800
English-Slovenian	Direct	9634	19,641,457	89,800
English-Slovenian	OES	182,787	19,641,457	294,318
English-Swedish	Direct	51,961	17,660,152	270,000
Estonian-Finnish	Direct	2191	9,504,879	12,800
Estonian-Finnish	Triang	43,313	9,504,879	78,200
Croatian-Slovenian	Direct	266	15,636,933	3400
Croatian-Slovenian	Triang	3669	15,636,933	31,600
Lithuanian-Latvian	Direct	2478	219,617	11,200
Lithuanian-Latvian	Triang	14,545	219,617	28,200

The sizes of dictionaries are reported in the number of word pairs, the sizes of parallel corpora in the number of matching contexts, and the sizes of resulting datasets in the number of matched words in matched sentence pairs. The Type column describes the dictionary creation approach; "direct" and "triang" denote that the dictionary was created directly from Wiktionary or with triangulation via English, respectively; "OES" stands for the Oxford English-Slovene dictionary

The dictionaries made using the wikt2dict tool are noisy, so we manually filtered them. We replaced the accented vowels with their non-accented variants in languages that do not use accented letters for vowels (e.g., Slovene and Russian). We removed the extra non-alphabetic characters, such as hash symbol, brackets, pipe, etc. We also removed all the entries which contained multiple-word terms. We leave the extension to the alignment of multi-word terms for further work.

We used direct bilingual dictionaries for all language pairs, where one of the languages was English. We used direct dictionaries and dictionaries created with the triangulation via English for all the other pairs (i.e., if neither language is English). For the English-Slovene pair, we also tested a large, high quality, manually created, proprietary Oxford English-Slovene dictionary.

4 Contextual alignments

This section describes the proposed methods for cross-lingual alignment of ELMo contextual embeddings. We start with the ELMoGAN method for nonlinear alignment. In Sect. 4.1, we describe the architecture of the proposed alignments, followed by their training in Sect. 4.2. Based on the constructed contextual alignment datasets, it is also possible to align contextual embeddings with classical linear mappings. We describe this approach in Sect. 4.3.

4.1 Architecture of ELMoGAN

The proposed ELMoGAN nonlinear cross-lingual contextual embedding alignment method uses GANs [12]. GANs consist of two connected neural models, a generator and a discriminator. The two models are trained simultaneously via an adversarial process. The discriminator attempts to discern whether the data passed to its input is real or fake (i.e., artificially generated). At the same time, the generator attempts to generate artificial data, which can fool the discriminator. GANs play a zero-sum game, where the discriminator's success means the generator's failure and vice versa. By simultaneously training both networks, they both improve. GANs are mostly used on images, where the described process can lead to compelling new generated images.

Following the success of GANs in neural machine translation [24] and cross-lingual embeddings alignment [9, 25], we propose a novel supervised nonlinear mapping method using bidirectional GANs. We based our contextual alignment method, called ELMoGAN, on the model of Fu et al. [25]. Contrary to Fu et al. [25], who only used their method with non-contextual fastText embeddings [2] to align sentences, we align contextual ELMo embeddings [3], which is only possible by constructing a special contextual mapping datasets, described in Sect. 3. As these datasets encode words in context, they are much larger and the resulting GANs have to more precisely map between cross-lingual vector spaces compared to non-contextual mappings.

The GAN-based cross-lingual embedding mapping comprises the generator module and discriminator module. The generator module (see Fig. 1) contains two generators that map vectors from one vector space to the other. Specifically, for a pair of languages L_1 and L_2, one generator will map from L_1 to L_2, and the second will map from L_2 to L_1. The two generators are completely independent of one another, and they do not share the data during training. The discriminator module (Fig. 2) contains two discriminators. The first discriminator (D_{valid}) tries to predict whether a given pair of vectors (l_1^x, l_2^y) represent the same token, i.e., if the first vector represents the word x in L_1 and the second vector represents the translation of the word x in L_2 (i.e., y). The second discriminator (D_{domain}) attempts to learn the difference between the direction of mapping. For a given pair of vectors, it predicts whether they are a vector from L_1 and its mapping to L_2 or a vector from L_2 and its mapping to L_1.

Compared to the ABSent model by Fu et al. [25] that was trained on a much smaller and easier problem of non-contextual mapping, in ELMoGAN we increased the size of all hidden layers in generators and discriminators. We also significantly lowered the learning rate as we achieved poor results with the learning rate used by Fu et al. [25]. The two generators in ELMoGAN have the same architecture: the input layer is followed by three fully connected feed-forward layers of 2048, 4096, and 2048 nodes. We used the ReLU activation function for all three layers. We added a batch normalization layer between each fully connected layer. The output layer has the same size as the input layer. It uses a hyperbolic tangent as the activation function so that the output is between -1 and $+1$. Both discriminators also have the same architecture. We first concatenate the two input vectors, then feed them to the three consecutive fully connected feed-forward layers with

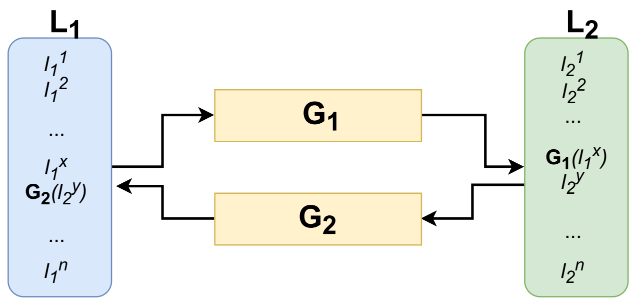


Fig. 1 The schema of the two generators for sentence alignment. Generator G_1 maps from L_1 to L_2 and generator G_2 does vice-versa. For a given pair of matching words x and y, where x is from L_1 and y from L_2, the generator G_1 attempts to map the vector l_1^x, so that its output $G_1(l_1^x)$ is as close as possible to l_2^y. The generator G_2 attempts to map l_2^y, so that $G_2(l_2^y)$ is as close as possible to l_1^x

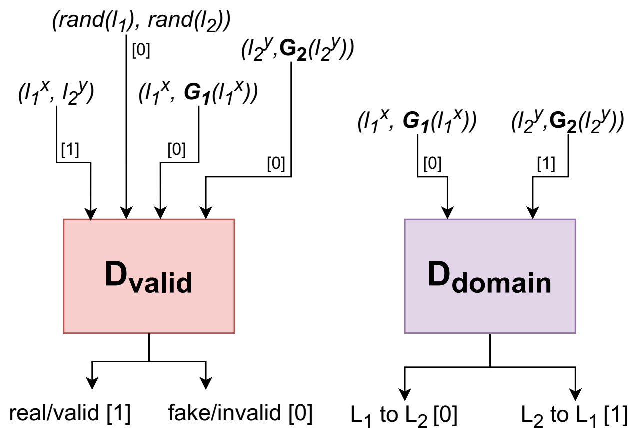


Fig. 2 The schema of the two discriminators for sentence alignment. Each discriminator receives a pair of vectors on input. During training D_{valid} receives a pair (l_1^x, l_2^y) labelled as a real pair, a pair of random word vectors labelled as a fake pair, and outputs from generators labelled as fake pairs. The D_{domain} discriminator receives outputs from generators on its input and attempts to discern between them

leaky ReLU ($\alpha = 0.2$). The output layer is a single neuron with sigmoid activation.

Typically, generators in GANs are optimized via the loss function, which compares the discriminator's output with the correct label. In our setup, generators did not successfully learn the mapping with this method, so we added another loss function, which aims to minimize the distance between l_1^x and $G_2(l_2^y)$, and between l_2^y and $G_1(l_1^x)$, as shown in Fig. 1. The total loss function is defined as:

$$L = \alpha L_G + \beta L_{D_{\text{valid}}} + \gamma L_{D_{\text{domain}}}$$

$$L_G = \frac{1}{\|Z\|} \sum_{(x,y)\in Z} \left[-S_{\cos}(l_2^y, G_1(l_1^x)) - S_{\cos}(l_1^x, G_2(l_2^y)) \right]$$

$$L_{D_{\text{valid}}} = -\frac{1}{\|Z\|} \sum_{(x,y)\in Z} \left[r_{(x,y)} \cdot \log(D_{\text{valid}}(G_1(l_1^x), G_2(l_2^y))) + (1 - r_{(x,y)}) \cdot \log(1 - D_{\text{valid}}(G_1(l_1^x), G_2(l_2^y))) \right]$$

$$L_{D_{\text{domain}}} = -\frac{1}{\|Z\|} \sum_{(x,y)\in Z} \left[r_{(x,y)} \cdot \log(D_{\text{domain}}(l_2^y, G_2(l_2^y))) + (1 - r_{(x,y)}) \cdot \log(1 - D_{\text{domain}}(l_1^x, G_1(l_1^x))) \right]$$

where $S_{\cos}$ is cosine similarity metric, and (x, y) is the pair of words from the contextual dataset Z, described in Sect. 3. Specifically, (x, y) is equivalent to (iD_j^A, iD_j^B), and (l_1^x, l_2^y) is equivalent to $(e(D_j^A, P_i^A), e(D_j^B, P_i^B))$. $r_{x,y}$ is the label (1 or 0) for the given pair of words (x, y) as displayed in Fig. 2. We set the factors α, β, and γ experimentally to $\frac{2}{12}$, $\frac{5}{12}$, and $\frac{5}{12}$, respectively. The results achieved using different α, β, and γ weights are shown in "Appendix C".

4.2 Training of ELMoGAN

We jointly trained the generator and discriminator modules using the parallel ELMo vectors datasets, described in Sect. 3. We trained ELMoGAN with the batch size of 256, Adam optimizer with the learning rate of 2×10^{-5}, and the learning rate decay of 10^{-5}. For each language pair, we trained three mapping models, one for each of the ELMo layers. For all three models, we used the same settings.

Each training iteration was done in two phases. In the first phase, we trained both discriminators independently. In the second phase, we trained the generator by training the whole GAN, while keeping the discriminator weights frozen.

To train the discriminators, we feed the discriminator D_{valid} four different types of vector pairs, as shown in Fig. 2. The first type are matching pairs (True), representing the same token in each respective language. Other types represent pairs of mismatching vectors (False). For the vector pairs labelled as True, we take the matching pairs from our training dataset. For the vector pairs labelled as False, we have three types of pairs. The first type is two randomly selected vectors from our dataset (one from each vector space). The second type is vectors from L_1 and their mappings, using the G_1 generator. The third type is vectors from L_2 and their mappings, using the G_2 generator. We chose to explicitly compare the mappings $G_1(l_1^x)$ and $G_2(l_2^y)$ with their original vectors (i.e., l_1^x and l_2^y, respectively) and not with each other. If we fed the discriminator a pair $(G_1(l_1^x), G_2(l_2^y))$, the model could learn to map each vector to an unrelated vector space. Additionally, by feeding the discriminator two randomly selected vectors, we wish to keep the two vector spaces disjunctive.

In the discriminator input pair (I_1, I_2), one of the vectors is from L_1 and the other is from L_2, but not always in the same order. During the discriminator training, the inputs are shown in Fig. 2, and in the second phase, the inputs to D_{valid} are $(G_1(l_1^x), G_2(l_2^y))$. For the first phase, we tested various combinations of the vector order, such as: I_1 always from L_2 and I_2 always from L_1; half of input pairs have I_1 from L_1 and I_2 from L_2, the other half has I_1 from L_2 and I_2 from L_1; etc. The best results among the tested combinations were achieved using the ordering shown in Fig. 2. The proportion of the four input types for D_{valid} was equal, that is, in each iteration, the number of vector pairs of each described type above was the same. We tested various proportions of the vector pairs and described the results in "Appendix C".

For the second phase of the training, we feed the generators the word vectors from our training dataset. On the input of the first generator are the vectors from L_1, and on the output, there are the matching vectors from L_2; vice-versa is true for the second generator.

We produced two different versions of ELMoGAN models, based on the number of iterations used in the training. The first version of models, called ELMoGAN-10k, were trained for a fixed number of 10,000 iterations for each layer of each language pair. The second version of models, called ELMoGAN-O, were trained models with several different numbers of iterations. We evaluated them on a dictionary induction task and selected the number of iterations that gave the best result. The details of selecting the number of iterations are presented in "Appendix A".

4.3 Cross-lingual linear mappings for contextual embeddings

Besides nonlinear mappings with GANs described above, it is also possible to compute cross-lingual mappings between contextual embeddings based on the standard assumption that the aligned spaces are largely isomorphic. This assumption may be approximately valid for similar languages. Below, we shortly describe methods for this type of alignments.

With a large enough collection of words in matching contexts (as described in Sect. 3), we compute their contextual embedding vectors and align them with any of the non-contextual mapping methods (described in Sect. 2.1). We use two such well-know and successful methods. The Vecmap methods [14] change both source and target embedding space, which makes them computationally less efficient on downstream tasks analyzed in Sect. 5.6. Methods from the MUSE library [9] only align source vectors to target vectors and are therefore computationally more efficient. As discussed in Sect. 2.4, a similar approach was proposed by Aldarmaki and Diab [27] but did not use large contextual datasets (presented in Sect. 3) based on high-quality dictionaries as we did.

5 Evaluation

In this section, we compare the four proposed alignment methods for contextual ELMo embeddings: two nonlinear ELMoGAN methods and two linear methods. We evaluated the methods on four downstream tasks: NER (Sect. 5.1), DP (Sect. 5.2), terminology alignment (Sect. 5.3), and sentiment analysis (Sect. 5.4). We compare two nonlinear methods, ELMoGAN-10k and ELMoGAN-O methods, trained as described in Sect. 4.2, with two linear mapping methods, MUSE [9] and Vecmap [8, 14], adapted for contextual embeddings, as described in Sect. 4.3. These two are linear cross-lingual mapping methods that assume linear transformation between vector spaces. For training of all alignments, we used the same datasets, described in Sect. 3. In all the experiments, we use embedding mappings from a target (i.e., evaluation) language to a train language; namely, we map the embeddings of the language used for the evaluation to the vector space of the language, which was used during the training of the model.

To better interpret the obtained results, we conducted two further ablation studies. In Sect. 5.5, we tested the importance of alignment dataset size. We used the English-Slovene pair, where we have available a large high-quality proprietary Oxford English-Slovene dictionary, instead of the publicly available Wiktionary. In Sect. 5.6, we tested different variants of the Vecmap alignment approach to check if we can avoid transforming both the source and target vector space and thereby significantly speed-up the approach.

5.1 Named entity recognition

NER is an information extraction task that seeks to locate and classify named entities mentioned in text into pre-defined categories such as person names, organizations, locations, medical codes, time expressions, quantities, monetary values, etc. In our experiments, the labels in the used NER datasets are simplified to a common label set of four labels present in all the addressed languages. These labels are "person", "location", "organization", and "other". The latter encompasses all named entities that do not fall in one of the three mentioned classes and all the tokens that are not named entities. The datasets used in the evaluation on the NER task are shown in Table 2, along with some basic statistics of the datasets.

In Table 3, we present the results using the Macro F_1 score, which is an average of F_1 scores for each class we are trying to predict, excluding the class "other" (i.e., not a named entity). The upper part of Table 3 shows a typical cross-lingual transfer learning scenario, where the model is transferred from resource-rich language (English) to less-resourced languages. In this case, the nonlinear ELMoGAN methods, in particular the ELMoGAN-10k variant, are superior to linear Vecmap and MUSE approaches. In this scenario, ELMoGAN-10k is always the best or close to the best mapping approach. This is not the case in the lower part of Table 3, which shows the second most important cross-lingual transfer scenario: transfer between similar languages. In this scenario, linear Vecmap and MUSE perform best in all twelve language pairs (seven times Vecmap and five times MUSE). We hypothesize that the reason for better performance of linear mappings is the similarity of tested language pairs and therefore lesser violation of the isomorphism assumption the Vecmap and MUSE method make. The results of the MUSE method support this hypothesis. While MUSE performs worst in most cases of transfer from English, the performance gap is smaller for transfer between similar languages. MUSE is sometimes the best method for similar languages, but the results of MUSE fluctuate greatly between language pairs. The second possible factor explaining the results is the quality of the dictionaries, which are in general better for combinations involving English. In particular, dictionaries obtained by triangulation via English are of poor quality, and nonlinear transformations might be more affected by the imprecision of anchor points.

In general (for NER and other tasks presented later), even the best cross-lingual prediction models lag behind the model without cross-lingual transfer (the column Direct in Table 3). The differences in Macro F_1 score are small for some languages (e.g., 5.5% for English-Swedish), but they are significantly larger for most of the languages.

Table 2 The datasets for the NER task and their properties: the number of sentences and number of tagged words

Language	Corpus	Sentences	Tags
Croatian	hr500k [33]	25,000	29,000
English	CoNLL-2003 NER [34]	21,000	44,000
Estonian	Estonian NER corpus [35]	14,000	21,000
Finnish	FiNER data [36]	14,500	17,000
Latvian	LV Tagger train data [37]	10,000	11,500
Lithuanian	TildeNER [38]	5476	7024
Slovene	ssj500k [39]	9500	9500
Swedish	Swedish NER [40]	8500	7500

5.2 Dependency parsing

The DP task constructs a dependency tree of a given sentence. In DP, all the words in a sentence are arranged into a hierarchical tree, based on their semantic dependencies. Each word has at most one parent node, and only the root word has no parent. A word can have multiple children nodes. In addition to predicting the tree's structure, the task is also to label the hierarchical dependencies.

As the DP architecture, we use the SuPar tool by Yu Zhang[10], which is based on the deep biaffine attention [41]. We modified the SuPar tool to accept ELMo embeddings on the input; specifically, we used the concatenation of the three ELMo layers. We made the modified code publicly available[11]. We trained the parser for 10 epochs, using datasets in nine languages (Croatian, English, Estonian,

[10] https://github.com/yzhangcs/parser.

[11] https://github.com/MatejUlcar/parser/tree/elmo.

Table 3 Comparison of different methods for cross-lingual mapping of contextual ELMo embeddings evaluated on the NER task

Source	Target	Dictionary	Vecmap	ELMoGAN-O	ELMoGAN-10k	MUSE	Direct
English	Croatian	Direct	**0.385**	0.274	0.365	0.024	0.810
English	Estonian	Direct	0.554	0.693	**0.728**	0.284	0.895
English	Finnish	Direct	0.672	0.705	**0.780**	0.229	0.922
English	Latvian	Direct	0.499	0.644	**0.652**	0.216	0.818
English	Lithuanian	Direct	0.498	0.522	**0.553**	0.208	0.755
English	Slovenian	Direct	0.548	0.572	**0.676**	0.060	0.850
English	Swedish	Direct	**0.786**	0.700	0.780	0.568	0.852
Croatian	Slovenian	Direct	0.387	0.279	0.250	**0.418**	0.850
Croatian	Slovenian	Triang	**0.731**	0.365	0.420	0.592	0.850
Estonian	Finnish	Direct	**0.517**	0.339	0.316	0.278	0.922
Estonian	Finnish	Triang	**0.779**	0.365	0.388	0.296	0.922
Finnish	Estonian	Direct	0.477	0.305	0.324	**0.506**	0.895
Finnish	Estonian	Triang	**0.581**	0.334	0.376	0.549	0.895
Latvian	Lithuanian	Direct	**0.423**	0.398	0.404	0.345	0.755
Latvian	Lithuanian	Triang	**0.569**	0.445	0.472	0.378	0.755
Lithuanian	Latvian	Direct	0.263	0.312	0.335	**0.604**	0.818
Lithuanian	Latvian	Triang	0.359	0.405	0.409	**0.710**	0.818
Slovenian	Croatian	Direct	0.361	0.270	0.307	**0.485**	0.810
Slovenian	Croatian	Triang	**0.566**	0.302	0.321	0.518	0.810
Average gap for the best cross-lingual transfer in each language							0.147

The best Macro F_1 score for each language pair is in bold. The "Direct" column represents the upper bound of achievable score using direct learning on the target language without cross-lingual transfer. The upper part of table contains a scenario of cross-lingual transfer from English to a less-resourced language, and the lower part of table shows a transfer between similar languages

Finnish, Latvian, Lithuanian, Russian, Slovene, and Swedish). The datasets are obtained from the Universal Dependencies [42] version 2.3. The datasets used and their basic statistics are shown in Table 4.

We used two evaluation metrics in the DP task, the unlabelled and labelled attachment scores (UAS and LAS) on the test set. The UAS and LAS are standard accuracy metrics in DP. The UAS score is defined as the proportion of tokens that are assigned the correct syntactic head. The LAS score is the proportion of tokens assigned the correct syntactic head and the correct dependency label [44].

The results in Table 5 show that the Vecmap and MUSE mapping methods outperform both ELMoGAN methods on most language pairs in this task. Larger dictionaries, created with triangulation, performed better than smaller direct dictionaries, despite the triangulated dictionaries being of worse quality. Language pairs with similar languages performed better than when the training language was English.

The MUSE method is stable on the DP task, which is not the case on the NER task. While MUSE outperforms Vecmap on a few language pairs, its results still lag behind Vecmap on average.

5.3 Terminology alignment

Terms are single words or multi-word expressions denoting concepts from specific subject fields (e.g., embedding or machine learning). The bilingual terminology alignment task aligns terms between two candidate term lists in two different languages. Given a pair of terms t_1 and t_2, where t_1 is from one language and t_2 is its equivalent from the second language, we measured the cosine distance between vector of t_1 and vectors of all terms from the second language. If the vector of t_2 is the closest to t_1 among all the terms from the second language, we count the pair as correctly aligned. For example, for a pair of terms from the Slovenian-English term bank "računovodstvo - accounting", we map the Slovene word embedding of the word "računovodstvo" from Slovene to English and check among all English word vectors for the vector that is the closest to the mapped Slovenian vector for "računovodstvo". If the closest vector is "accounting", we count this as a success, else as a failure. This measure is called accuracy@1 score or 1NN score and is defined as the number of successes divided by the number of all examples, in this case term pairs.

We extracted aligned bilingual term lists from Eurovoc [45], a multilingual thesaurus with more than 10,000 terms

Table 4 Dependency parsing datasets and their properties: the treebank, language ISO 639-1 code, number of sentences, number of tokens, and information about the size of the split

Language	Code	Treebank	Tokens	Sentences	Train	Validation	Test
Croatian	hr	SET	199,409	9010	6914	960	1136
English	en	EWT	254,855	16,622	12,543	2002	2077
Estonian	et	EDT	438,171	30,972	24,633	3125	3214
Finnish	fi	TDT	202,697	15,135	12,216	1364	1555
Latvian	lv	LVTB	220,536	13,643	10,156	1664	1823
Lithuanian	lt	ALKSNIS [43]	70,051	3642	2341	617	684
Russian	ru	GSD	98,000	5030	3850	579	601
Slovene	sl	SSJ	140,670	8000	6478	734	788
Swedish	sv	Talbanken	96,858	6026	4303	504	1219

Table 5 Comparison of different contextual cross-lingual mapping methods on the DP task

Train	Eval.		Vecmap		ELMoGAN-O		ELMoGAN-10k		MUSE		Direct	
lang.	lang.	Dict.	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
en	hr	Direct	**73.96**	**60.53**	68.73	50.29	66.74	40.93	71.01	54.89	91.74	85.84
en	et	Direct	**62.08**	**40.62**	52.01	30.22	44.80	24.59	58.76	34.07	89.54	85.45
en	fi	Direct	**64.40**	**45.32**	50.80	25.23	42.65	22.66	55.03	37.61	90.83	86.86
en	lv	Direct	**77.84**	**65.97**	68.51	49.47	67.09	39.41	76.26	63.45	88.85	82.82
en	lt	Direct	**63.33**	**40.56**	50.04	31.26	49.22	25.15	58.70	37.78	82.84	72.16
en	ru	Direct	**72.00**	**16.62**	60.74	8.92	60.68	8.18	65.23	14.77	89.33	83.54
en	sl	Direct	**79.01**	**59.84**	68.82	48.20	67.04	43.34	77.18	56.53	93.70	91.39
en	sv	Direct	82.08	72.74	74.39	59.70	73.81	59.63	**82.17**	**72.78**	89.70	85.07
hr	sl	Direct	**85.47**	**72.70**	51.88	31.50	53.68	33.40	83.45	69.08	93.70	91.39
hr	sl	Triang	**87.70**	**76.51**	54.34	36.32	59.61	38.83	**87.70**	76.40	93.70	91.39
et	fi	Direct	**79.14**	**66.09**	55.67	36.85	51.35	30.66	76.66	60.01	90.83	86.86
et	fi	Triang	**80.94**	**67.35**	52.63	29.94	52.83	28.70	76.96	63.37	90.83	86.86
fi	et	Direct	**75.81**	57.32	54.69	33.99	53.27	32.28	74.96	**58.14**	89.54	85.45
fi	et	Triang	**79.04**	**61.86**	53.64	32.73	53.86	30.13	76.74	60.27	89.54	85.45
lv	lt	Direct	**76.43**	**54.24**	64.44	37.16	64.73	35.86	75.45	53.02	82.84	72.16
lv	lt	Triang	**76.26**	**53.59**	65.91	37.91	65.45	33.62	75.12	51.14	82.84	72.16
lt	lv	Direct	63.27	24.53	56.43	26.93	62.51	31.84	**73.70**	**44.62**	88.85	82.82
lt	lv	Triang	61.32	27.29	61.89	29.39	61.95	30.11	**72.39**	**43.15**	88.85	82.82
sl	hr	Direct	**77.89**	**62.58**	47.34	29.39	52.27	32.48	72.87	55.70	91.74	85.84
sl	hr	Triang	**81.32**	**67.51**	50.96	32.82	56.17	35.96	78.63	63.96	91.74	85.84
Avg. gap for the best cross-lingual transfer in each language											9.89	23.79

Results are reported as unlabelled attachments score (UAS) and labelled attachment score (LAS). The column “Direct“ stands for direct learning on the target (i.e., evaluation) language without cross-lingual transfer. The languages are represented with their international language codes ISO 639-1

available in all EU languages. For building contextualised vector representations of these terms, we used the Europarl corpus [46, 47]. For Croatian, Europarl is not available, so we used DGT translation memory [48] instead.

For each word, we concatenate the three ELMo vectors into one 3072-dimensional vector. For single-word terms, we represent each term as the average vector of all contextual vector representations for that word, found in the corpus. For multi-word terms, we used a two-step approach. First, we check whether the term appears in the corpus. If it does, we represent each term occurrence as the average vector of the words it is composed of. We then average over all the occurrences, as with single-word terms. If the term does not appear in the corpus, we represent it as the average of all words it is composed of, where word vectors are averaged over all occurrences in the corpus. For each language pair, we evaluate the terminology alignment in both directions. That is, given the terms from the first language (source), we search for the equivalent terms in the second language (target), then we repeat the process in the other direction.

We present the results of cross-lingual terminology alignment of contextual ELMo embeddings in Table 6. We compared the same four mapping methods as in the previous tasks. For terminology alignment between English and other languages, the two nonlinear mappings perform the best on all language pairs. With English as the target language, ELMoGAN-10k always performs the best. In cases where English is the source language, ELMoGAN-O is usually the best.

Table 6 Comparison of contextual cross-lingual mapping methods for ELMo embeddings, evaluated on the terminology alignment task

Source	Target	Dict (Dir)	Vecmap	EG-O	EG-10k	MUSE
en	sl	Direct (sl→en)	0.079	**0.152**	0.151	0.096
sl	en	Direct (sl→en)	0.099	0.139	**0.195**	0.126
en	hr	Direct (hr→en)	0.080	**0.153**	0.135	0.116
hr	en	Direct (hr→en)	0.084	0.139	**0.153**	0.102
en	et	Direct (et→en)	0.092	**0.177**	0.167	0.128
et	en	Direct (et→en)	0.091	0.117	**0.133**	0.118
en	fi	Direct (fi→en)	0.092	0.166	**0.176**	0.132
fi	en	Direct (fi→en)	0.087	0.083	**0.116**	0.112
en	lv	Direct (lv→en)	0.084	**0.157**	0.147	0.102
lv	en	Direct (lv→en)	0.091	0.122	**0.140**	0.111
en	lt	Direct (lt→en)	0.095	**0.181**	0.172	0.114
lt	en	Direct (lt→en)	0.097	0.132	**0.171**	0.102
en	sv	Direct (sv→en)	0.125	0.183	**0.187**	0.161
sv	en	Direct (sv→en)	0.112	0.111	**0.167**	0.109
sl	hr	Direct (hr→sl)	**0.109**	0.037	0.031	0.102
sl	hr	Triang (hr→sl)	0.130	0.056	0.046	**0.156**
sl	hr	Direct (sl→hr)	**0.109**	0.039	0.038	0.100
sl	hr	Triang (sl→hr)	0.130	0.053	0.057	**0.155**
hr	sl	Direct (hr→sl)	**0.084**	0.029	0.028	0.082
hr	sl	Triang (hr→sl)	0.097	0.042	0.044	**0.121**
hr	sl	Direct (sl→hr)	**0.084**	0.023	0.021	**0.084**
hr	sl	Triang (sl→hr)	0.097	0.039	0.033	**0.121**
fi	et	Direct (et→fi)	**0.130**	0.092	0.078	0.121
fi	et	Triang (et→fi)	**0.130**	0.102	0.080	0.124
fi	et	Direct (fi→et)	**0.129**	0.085	0.089	0.122
fi	et	Triang (fi→et)	0.130	0.090	0.094	**0.145**
et	fi	Direct (et→fi)	0.143	0.091	0.094	**0.167**
et	fi	Triang (et→fi)	0.148	0.095	0.103	**0.166**
et	fi	Direct (fi→et)	0.143	0.108	0.092	**0.166**
et	fi	Triang (fi→et)	0.148	0.118	0.097	**0.189**
lv	lt	Direct (lt→lv)	0.102	0.080	0.061	**0.123**
lv	lt	Triang (lt→lv)	0.119	0.090	0.076	**0.134**
lv	lt	Direct (lv→lt)	0.102	0.059	0.071	**0.123**
lv	lt	Triang (lv→lt)	0.119	0.065	0.077	**0.128**
lt	lv	Direct (lt→lv)	0.099	0.061	0.069	**0.102**
lt	lv	Triang (lt→lv)	0.112	0.064	0.076	**0.116**
lt	lv	Direct (lv→lt)	0.099	0.071	0.057	**0.102**
lt	lv	Triang (lv→lt)	**0.112**	0.083	0.069	0.110

Results are reported as accuracy@1, based on the cosine distance metric. The best results for each language and type of transfer (from English in the top part and from similar language the bottom part of table) are typeset in **bold**. The languages are represented with their international language codes ISO 639-1. The labels "Dict" and "Dir" in the third column represent the type of the dictionary and the direction of vector mappings: from→to

For terminology alignment between similar languages, we also compared the mapping of embeddings in both directions, that is we mapped source terms embeddings to target terms space and vice-versa. Linear methods outperform the nonlinear methods on similar languages. In most cases, MUSE is the best method. If we just look at the best dictionary and mapping direction for each language pair, MUSE is the best in each language pair not involving English. The terminology alignment is in most cases better from English than from a similar language as a source, the exceptions are Croatian and Finnish (as targets).

Overall, it seems that results on the terminology alignment task follow the same pattern as in the NER task: dominance of nonlinear methods in the transfer from English and prevalence of linear methods in the transfer from similar languages.

5.4 Sentiment analysis

We treat the sentiment analysis task as a sentence-level classification. Given a sentence or a short text of a few sentences, we label it with one of the three predefined categories: positive, negative, or neutral, based on the text's sentiment. We used large Twitter sentiment datasets by Mozetič et al. [49], extracting the four languages we are analysing in other tasks, i.e., English, Croatian, Slovenian, and Russian.

We trained the classifiers using the same approach and hyper-parameters as for NER (described in Sect. 5.1), but with a different neural network architecture. We trained models with four hidden layers, three bidirectional LSTMs and one fully connected feed-forward layer. The three LSTM layers have 512, 512, and 256 units, respectively. The fully connected layer has 64 neurons.

We present the results using the macro F_1 score in Table 7. The difference between transfer from English and transfer from a similar language is much smaller in this task. Similarly, the performance between direct dictionaries and those created using triangulation is not large in most cases. Triangulated dictionaries perform better overall, but not for every mapping method. MUSE method outperforms the other three methods in all, but one example, transfer from English to Croatian, where all methods perform on par, but MUSE slightly worse than the other three. For transfer from Slovenian to Croatian, both ELMoGAN methods show large variance between individual runs and outperform MUSE on some runs. However, averaged over several runs they perform significantly worse.

Overall, the results on the sentiment analysis task follow the same pattern as on the DP task: dominance of the linear mapping methods.

5.5 Dataset size importance

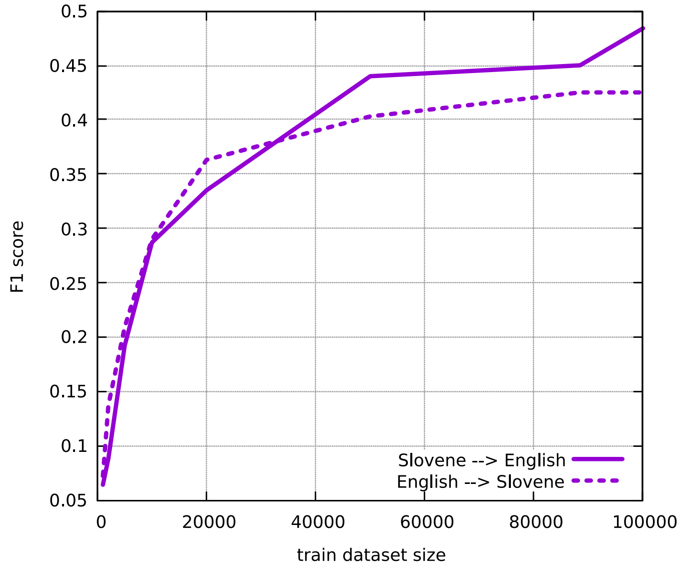
We tested the importance of dataset size on the English-Slovene language pair. In the contextual dataset creation, we used a large, high-quality Oxford English-Slovene dictionary instead of Wiktionary. We kept all the other resources and settings the same. We evaluated ELMoGAN-10k on NER and DP tasks using various sizes of the dataset to train contextual alignments. One of the dataset sizes is 89800 entries, which is the same size as the dataset created with the low-quality Wiktionary dictionary. We included that size for easier comparison between both dictionaries.

The results on the NER task are shown in Fig. 3. When we increase the size of the dataset, the performance on the NER task improves. The dataset size matters, and we presume that the performance would further increase with an even larger dataset. Surprisingly, the results achieved with the dataset of size 89800 are significantly worse than the results achieved with the dataset of the same size, created with the low-quality dictionary (see Table 3). Using the Oxford English-Slovene dictionary, we achieved the F_1 score of 0.450 when trained on English and evaluated on Slovene. Using Wiktionary bilingual dictionary, we

Table 7 Comparison of different methods for cross-lingual mapping of contextual ELMo embeddings, evaluated on the sentiment analysis task

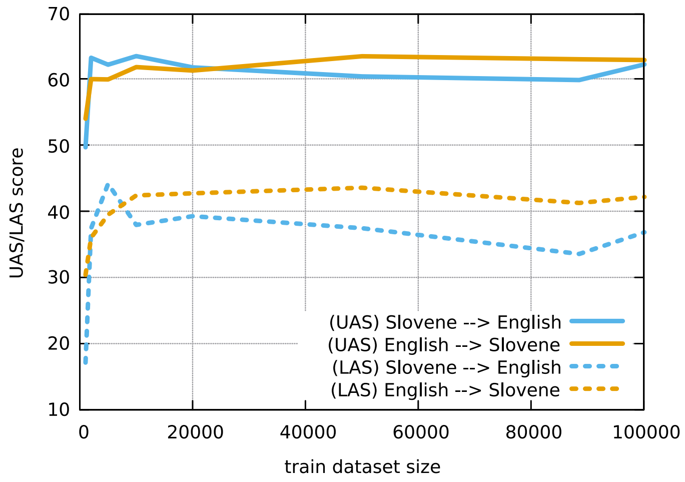
Source	Target	Dict.	Vecmap	ELMoGAN-O	ELMoGAN-10k	MUSE	Direct
en	hr	Direct	**0.44**	**0.44**	**0.44**	0.43	0.59
en	sl	Direct	0.45	0.45	0.46	**0.47**	0.51
en	ru	Direct	0.45	0.35	0.37	**0.48**	0.68
hr	sl	Direct	0.45	0.39	0.39	**0.48**	0.51
hr	sl	Triang	0.46	0.40	0.41	**0.47**	0.51
sl	hr	Direct	0.33	0.28	0.36	**0.48**	0.59
sl	hr	Triang	0.47	0.29	0.31	**0.50**	0.59

The best macro F_1 score for each language pair is in bold. The "Direct" column represents direct learning on the target language without cross-lingual transfer. The upper part of table contains a scenario of cross-lingual transfer from English to a less-resourced language, and the lower part of table shows a transfer between similar languages



Train	Eval.	size	macro F_1
en	sl	1k	0.064
en	sl	2k	0.088
en	sl	5k	0.192
en	sl	10k	0.287
en	sl	20k	0.335
en	sl	50k	0.440
en	sl	100k	**0.484**
en	sl	LQsize	0.450
sl	en	1k	0.072
sl	en	2k	0.137
sl	en	5k	0.209
sl	en	10k	0.290
sl	en	20k	0.363
sl	en	50k	0.403
sl	en	100k	**0.425**
sl	en	LQsize	**0.425**

Fig. 3 Comparison of different sizes of cross-lingual contextual datasets based on different dictionaries used for cross-lingual mapping of contextual ELMo embeddings, evaluated on the NER task. LQsize represents the size of the dataset equal to the size of the low-quality dictionary (89,800 total entries, 88,453 entries in the training dataset). The mapping method used was ELMoGAN-10k



Train	Eval.	size	UAS	LAS
en	sl	1k	49.71	17.05
en	sl	2k	63.28	37.40
en	sl	5k	62.25	**44.16**
en	sl	10k	**63.53**	37.94
en	sl	20k	61.83	39.26
en	sl	50k	60.47	37.43
en	sl	100k	62.28	36.81
en	sl	LQsize	59.90	33.53
sl	en	1k	54.06	30.35
sl	en	2k	60.05	36.00
sl	en	5k	60.00	39.49
sl	en	10k	61.89	42.41
sl	en	20k	61.34	42.73
sl	en	50k	**63.53**	**43.58**
sl	en	100k	62.94	42.18
sl	en	LQsize	63.07	41.27

Fig. 4 Comparison of different sizes of cross-lingual contextual datasets based on different dictionaries used for cross-lingual mapping of contextual ELMo embeddings, evaluated on the DP task. LQsize represents the size of the dataset based on the low-quality dictionary (89800 total entries, 88453 entries in train dataset). We used the ELMoGAN-10k mapping method

achieved the F_1 score of 0.676 on the same language pair with the same alignment method.

The results in the DP task show different behavior. At first, the performance quickly increases with larger datasets, and then it stays the same or even starts to drop (see Fig. 4). The best results are achieved with the dataset of size 50,000 when mapping from English to Slovene. When mapping from Slovene to English, datasets of size only 10,000 (based on the UAS) and 5000 (based on the LAS) produce the best results.

The better performance of Wiktionary bilingual dictionary over high-quality Oxford dictionary, when datasets are of the same size, is observed in the DP task as well. On Slovene to English mapping, the dataset from Oxford dictionary scores 59.90% UAS and 33.53% LAS.

Wiktionary-based dataset scores 67.04% UAS and 43.34% LAS.

The results on dataset size and results from Sects. 5.1 and 5.2 lead us to the conclusion that the quality of the dictionary used does not play a large role. The more important parameter is the size of the dictionary. On the NER task, larger dictionary sizes always improve results. The DP task results remain inconclusive, as the larger dictionary created with triangulation outperformed the smaller direct dictionary for similar language pairs on the DP task.

5.6 Vecmap optimizations

In computing cross-lingual alignments of two languages, the Vecmap method changes both embedding spaces. This means that we have to train a separate embedding for each language pair. In our case, we had to train eight different English models on English data for each downstream task, one for each pair of languages, when using Vecmap for alignments. The reason is that the English vectors change during the alignment as well, and we have to apply that change at the time of classifier model training. This considerably slows down the training and evaluating procedure. We tested several approaches to avoid retraining separate models, but none was successful. A detailed description of our experiments is contained in "Appendix B".

6 Conclusion

We present four novel methods for cross-lingual mapping of contextual ELMo embeddings. The two ELMoGAN methods (ELMoGAN-O and ELMoGAN-10k) do not assume isomorphic embedding spaces and use GANs to compute the alignments. The two linear methods use Vecmap and MUSE libraries, but map words in matching contexts. To construct the contextual mappings, all four methods need contextual embeddings datasets. We constructed fifteen such datasets for eleven language pairs. We created a matching set of contextual word embeddings for each language pair and each ELMo layer from parallel corpora and bilingual dictionaries.

The presented methods can be adapted for cross-lingual alignments of other contextual (and non-contextual) embeddings with a few caveats. As the methods align word vectors prior to their usage in a downstream model, this precludes their usage in models which are fully fine-tuned on a downstream task (e.g., a typical use of BERT models). For embeddings other than ELMo, the architecture of the generator and discriminators presented in Sect. 4.1 should be adapted to the size of the embedding vectors.

Table 8 The number of iterations ELMoGAN-O was trained for, for each embedding layer and language pair

Language 1	Language 2	Dictionary	Layer 1	Layer 2	Layer 3
English	Croatian	Direct	15000	50000	30000
English	Estonian	Direct	12000	50000	40000
English	Finnish	Direct	40000	50000	50000
English	Latvian	Direct	10000	40000	50000
English	Lithuanian	Direct	30000	50000	40000
English	Slovenian	Direct	30000	50000	25000
English	Swedish	Direct	50000	50000	50000
Croatian	Slovenian	Direct	15000	40000	10000
Croatian	Slovenian	Triangular	25000	50000	25000
Estonian	Finnish	Direct	30000	25000	15000
Estonian	Finnish	Triangular	30000	50000	20000
Latvian	Lithuanian	Direct	25000	40000	30000
Latvian	Lithuanian	Triangular	50000	30000	25000

The optimal number of iterations was determined on the dictionary induction task

Furthermore, a different set of training hyper-parameters should be used for method to converge and learn optimal alignment. For non-contextual embeddings, the creation of contextual dataset presented in Sect. 3 is not necessary, otherwise the alignment methods can be used with the same alterations described for the contextual embeddings.

Nonlinear ELMoGAN methods outperformed linear mappings on the NER and terminology alignment, especially on distant languages where the assumption of isomorphic embedding spaces is strongly violated. The nonlinear methods performed worse on the DP and sentiment analysis tasks. There are substantial differences between languages and we advise users to test both linear and nonlinear methods on a specific task and language. The ELMoGAN approach is sensitive to the values of training

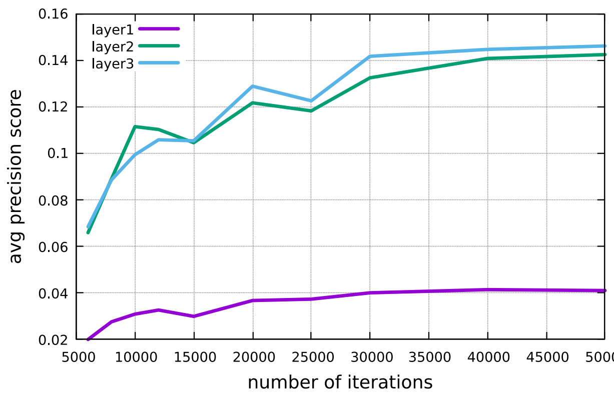


Fig. 5 The average precision score on dictionary induction task for English-Finnish alignment at different numbers of iterations of alignment algorithm

parameters, mostly the learning rate and the number of iterations. To find a set of well-performing hyper-parameters, the method has to be fine-tuned for each task.

There are still some open questions in practical application of ELMoGAN approach. First, it is not clear what is the best methodology for choosing the right number of iterations for each task. The dictionary induction task we currently use internally to determine the right number of iterations works well for the NER and terminology alignment tasks, but seems inappropriate for the DP task where greater emphasis is on syntactic properties of the language (and not so much on the words as in the NER task).

In further work, we intend to work on a robust method to find hyper-parameters. Testing more GAN architectures to find a more robust mapping might be a promising research direction. An issue worth investigating is multiple-word terms which are not included in the current contextual mapping datasets but could be useful in tasks requiring their correct cross-lingual recognition.

Appendix A: Tuning the number of iterations of ELMoGAN-O

ELMoGAN mapping models have been trained for a different number of iterations for each language pair and each ELMo layer. We have trained all models for the numbers of iterations set between 6000 and 50,000. We evaluated each model on the dictionary induction task on the evaluation part of our contextual mapping dataset, presented in Sect. 3. We have used the average score of precision@1, precision@5, and precision@10 for both directions of our bidirectional mapping model (i.e., from the first to the second language and reverse). The number of iterations that produced the best result on the evaluation set was selected as the optimal and was used in the model called ELMoGAN-O in other evaluations. The selected numbers of iterations are presented in Table 8.

We opted not to check more than 50000 iterations because the precision on the evaluation task rises quite quickly and then saturates or drops. For example, on the English-Finnish pair, the selected iterations were 40,000 for layer 1 and 50000 for layers 2 and 3. Still, these numbers do not fully reflect the optimal behavior for all languages and dictionaries. The precision scores for different iterations for this language pair are shown in Fig. 5.

Appendix B: Vecmap speed-up experiments

As explained in Sect. 5.6, the Vecmap method changes both languages' embedding spaces when computing the cross-lingual alignments. This means that we have to train a separate embedding for each language pair. In our case, we had to train eight different English models, one for each pair of languages. This considerably slows down the training and evaluating procedure. We tested six different sets of options for the Vecmap method to avoid retraining separate models. In this experiment, the training language was always English and we used the DP task. By default, Vecmap first normalizes both vector sets of a language pair and then calculates the mapping matrix, which maps vectors from one language to the other language (in our experiment, from each language to English). Finally, it re-weighs both sets of vectors. In the results below, we denote this approach as "ELMoVM" and it is identical to how we used the Vecmap method elsewhere in this paper. We tested five alternative approaches on the DP task; all of them were unsuccessful. This extra step of mapping both source and target languages seems to be unavoidable. The results for all the approaches are shown in Table 9.

The options for all the approaches are summarized in Table 10. The approach "et" is identical to ELMoVM, except that we used the English model trained for Estonian-English pair for all language pairs. The following four approaches do not alter English vectors in any way.

Table 9 Various options used with the Vecmap method on the DP task

Eval.	ELMoVM		et		orth		nonorm		evalnorm		def	
lang.	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
hr	**73.96**	**60.53**	26.79	1.83	13.22	1.14	17.03	2.29	25.67	6.10	16.54	0.72
et	**62.08**	**40.62**	62.08	40.62	11.97	1.21	9.38	0.76	20.05	1.62	13.82	1.53
fi	**64.40**	**45.32**	11.38	0.76	15.41	0.49	18.11	0.62	24.50	1.94	18.53	0.83
lv	**77.84**	**65.97**	25.32	2.21	12.82	1.26	12.88	0.63	29.10	7.89	20.39	1.96
lt	**67.92**	**39.62**	9.43	0.00	7.55	0.00	7.55	0.00	15.09	0.00	11.32	1.89
sl	**79.01**	**59.84**	28.92	2.53	13.72	0.87	12.06	0.48	25.22	6.45	14.33	0.83
sv	**82.08**	**72.74**	26.23	5.23	13.50	1.46	11.27	0.81	26.45	11.92	15.22	1.41

Train language is always English

Table 10 Options used (y) or not used (n) for different variants of Vecmap mapping

Method	ELMoVM	et	Orth	Nonorm	Evalnorm	Def
Train lang. mapped	Y	Y	N	N	N	N
Normalization at train time	Y	Y	N	N	N	N
Eval lang. mapped	Y	Y	Y	Y	Y	Y
Normalization at eval time	Y	Y	N	N	Y	Y
Normalization used for mapping calc.	Y	Y	Y	N	N	Y

Approach "orth" removes the normalization performed during the evaluation, but the normalization was still used for both languages to calculate the mapping matrix. Method "nonorm" is identical to "orth", except that we removed the normalization also during the mapping matrix calculation. Method "evalnorm" adds the normalization during the evaluation but does not use it during the mapping matrix calculation. Finally, the approach "def" uses the normalization both during the evaluation and mapping matrix calculation.

Appendix C: Ablation studies

The loss function of the ELMoGAN model, defined in Sect. 4.1 contains three tunable parameters, α, β, and γ:

$$L = \alpha L_{\text{gen}} + \beta L_{\text{Dvalid}} + \gamma L_{\text{Ddomain}}$$

We show that the first factor of the loss function above is important to successfully train the model. We evaluated eight different sets of weights α, β, and γ (Table 11) on the dictionary induction task, in the same manner as explained in "Appendix A", and on the NER task for the Slovenian-English pair.

On the same two tasks we also evaluated different approaches to training the D_{valid} discriminator, explained in Sect. 4.2. Specifically, the various amounts of each type of vector pairs it is trained on. We present the eleven different combinations tested in Table 12.

The results for the different loss function weights from Table 11 and inputs of the D_{valid} discriminator from Table 12 are shown in Table 13. We tested the alignment between English and Slovene ELMo embeddings. The results for the dictionary induction task are given as the average of P@1, P@5 and P@10 in both directions (English to Slovene, and Slovene to English). The results on the NER task show the performance of the NER model, trained on English data and evaluated on a portion of the Slovene train dataset, using Slovene ELMo embeddings and ELMoGAN-10k alignment model. At no point during the training of the English NER model or the alignment model was the Slovene NER train dataset seen.

The upper third of Table 13 shows the results for various weight sets from Table 11, while keeping every other setting unchanged (i.e., same as used in Sect. 5). The middle part of Table 13 shows the results for various sizes of inputs of the D_{valid} discriminator, while keeping the loss function weights the same as in Sect. 5, i.e., set A from Table 11. In the upper part of Table 13, we see that set D performs the best on average. In the bottom part of table, we show the results for various sizes of inputs of the D_{valid} discriminator, but using the loss function weights set D. Although we can see that some settings achieve poor results, there is no clear best set of parameters. There does not appear to be any correlation between the results of dictionary induction task and the results of the NER task, as a certain set of parameters may perform the best on one task, but poorly on another.

Table 11 Various sets of loss function weights

Weight set	α	β	γ
A	0.167	0.417	0.417
B	0.333	0.333	0.333
C	0.500	0.250	0.250
D	0.667	0.167	0.167
E	0.500	0.500	0.000
F	0.500	0.000	0.500
G	0.000	0.500	0.500
H	0.083	0.458	0.458

Table 12 Relative sizes of different inputs of the D_{valid} discriminator

#	x, y	rand_1,rand_2	x, $G(x)$	y, $G(y)$
1	1	1	1	1
2	1	1	1	0
3	1	1	0	1
4	1	0	1	1
5	1	1	0	0
6	1	0	1	0
7	1	0	0	1
8	1	0.5	0.5	0
9	1	0	0.5	0.5
10	1	0.5	0	0.5
11	1	0.333	0.333	0.333

Table 13 Dictionary induction results (average of P@1, P@5 and P@10) and NER results for different loss function weights and D_{valid} discriminator inputs

Parameters	CNN layer	LSTM 1	LSTM 2	NER
A1	0.294	0.236	0.267	0.660
B1	**0.317**	0.290	0.286	0.636
C1	0.307	0.314	0.299	0.640
D1	0.312	0.328	0.301	0.621
E1	0.271	0.342	0.292	0.486
F1	0.303	0.293	0.290	0.648
G1	0.002	0.000	0.000	0.019
H1	0.267	0.175	0.253	0.646
A1	0.294	0.236	0.267	0.660
A2	0.295	0.277	0.278	0.616
A3	0.293	0.246	0.269	**0.672**
A4	0.297	0.354	0.286	0.665
A5	0.296	0.297	0.274	0.632
A6	0.283	0.339	0.281	0.546
A7	0.303	0.367	0.351	0.546
A8	0.297	0.286	0.276	0.527
A9	0.290	0.339	0.289	0.614
A10	0.294	0.253	0.277	**0.672**
A11	0.310	0.261	0.274	0.656
D1	0.312	0.328	0.301	0.621
D2	0.284	0.336	0.294	0.537
D3	0.297	0.327	0.294	0.649
D4	0.299	0.361	0.313	0.649
D5	0.283	0.338	0.291	0.553
D6	0.276	0.342	0.295	0.491
D7	0.303	**0.377**	**0.367**	0.517
D8	0.283	0.336	0.292	0.536
D9	0.299	0.361	0.314	0.659
D10	0.310	0.334	0.295	0.610
D11	0.309	0.330	0.301	0.623

The best result in each of the three parts (upper, middle, lower) of table is underlined, the best result overall is in bold

Acknowledgements The work was partially supported by the Slovenian Research Agency (ARRS) core research programme P6-0411 and project J1-2480. This paper is supported by European Union's Horizon 2020 research and innovation programme under grant agreement No 825153, project EMBEDDIA (Cross-Lingual Embeddings for Less-Represented Languages in European News Media).

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Mikolov T, Le QV, Sutskever I (2013) Exploiting similarities among languages for machine translation. arXiv preprint 13094168
2. Bojanowski P, Grave E, Joulin A, Mikolov T (2017) Enriching word vectors with subword information. Trans Assoc Comput Ling 5:135–146
3. Peters M, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, Zettlemoyer L (2018) Deep contextualized word representations. In: Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: human language technologies, Volume 1 (Long Papers), pp 2227–2237
4. Devlin J, Chang MW, Lee K, Toutanova K (2019) BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, Volume 1 (Long and Short Papers), pp 4171–4186
5. Škvorc T, Gantar P, Robnik-Šikonja M (2022) MICE: mining idioms with contextual embeddings. Knowl Based Syst 235
6. Artetxe M, Schwenk H (2019) Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. Trans Assoc Comput Ling 7:597–610
7. Robnik-Šikonja M, Reba K, Mozetič I (2021) Cross-lingual transfer of sentiment classifiers. Slovenščina 2.0 9(1):1–25, https://doi.org/10.4312/slo2.0.2021.1.1-25
8. Artetxe M, Labaka G, Agirre E (2018) A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In: Proceedings of the 56th annual meeting of the association for computational linguistics (Volume 1: Long Papers), pp 789–798
9. Conneau A, Lample G, Ranzato M, Denoyer L, Jégou H (2018) Word translation without parallel data. In: Proceedings of international conference on learning representation (ICLR)
10. Ormazabal A, Artetxe M, Labaka G, Soroa A, Agirre E (2019) Analyzing the limitations of cross-lingual word embedding mappings. In: Proceedings of the 57th annual meeting of the association for computational linguistics ACL, pp 4990–4995
11. Søgaard A, Ruder S, Vulić I (2018) On the limitations of unsupervised bilingual dictionary induction. In: Proceedings of the 56th annual meeting of the association for computational linguistics (Volume 1: Long Papers), pp 778–788
12. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: Advances in neural information processing systems, pp 2672–2680
13. Søgaard A, Vulić I, Ruder S, Faruqui M (2019) Cross-lingual word embeddings. Morgan & Claypool Publishers, San Rafael
14. Artetxe M, Labaka G, Agirre E (2018) Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations. In: Thirty-second AAAI conference on artificial intelligence
15. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
16. Rogers A, Kovaleva O, Rumshisky A (2020) A primer in BERTology: what we know about how BERT works. Trans Assoc Comput Ling 8:842–866
17. Puccetti G, Miaschi A, Dell'Orletta F (2021) How do BERT embeddings organize linguistic knowledge? In: Proceedings of deep learning inside out (DeeLIO): the 2nd workshop on knowledge extraction and integration for deep learning architectures, pp 48–57
18. Peters ME, Ruder S, Smith NA (2019) To tune or not to tune? Adapting pretrained representations to diverse tasks. In:

Proceedings of the 4th workshop on representation learning for NLP (RepL4NLP-2019), pp 7–14

19. Krone J, Zhang Y, Diab M (2020) Learning to classify intents and slot labels given a handful of examples. In: Proceedings of the 2nd workshop on natural language processing for conversational AI, pp 96–108
20. Nakashole N, Flauger R (2018) Characterizing departures from linearity in word translation. In: Proceedings of the 56th annual meeting of the association for computational linguistics (Volume 2: Short Papers), pp 221–227
21. Lu A, Wang W, Bansal M, Gimpel K, Livescu K (2015) Deep multilingual correlation for improved word embeddings. In: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp 250–256
22. Andrew G, Arora R, Bilmes J, Livescu K (2013) Deep canonical correlation analysis. In: International conference on machine learning, pp 1247–1255
23. Zhao J, Gilman A (2020) Non-linearity in mapping based cross-lingual word embeddings. In: Proceedings of the 12th language resources and evaluation conference, pp 3583–3589
24. Yang Z, Chen W, Wang F, Xu B (2018) Improving neural machine translation with conditional sequence generative adversarial nets. In: Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: human language technologies, Volume 1 (Long Papers), pp 1346–1355
25. Fu Z, Xian Y, Geng S, Ge Y, Wang Y, Dong X, Wang G, de Melo G (2020) ABSent: cross-lingual sentence representation mapping with bidirectional GANs. In: Proceedings of the AAAI conference on artificial intelligence pp 7756–7763
26. Schuster T, Ram O, Barzilay R, Globerson A (2019) Cross-lingual alignment of contextual word embeddings, with applications to zero-shot dependency parsing. In: Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, Volume 1 (Long and Short Papers), pp 1599–1613
27. Aldarmaki H, Diab M (2019) Context-aware cross-lingual mapping. In: Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, Vol 1 (Long and Short Papers), pp 3906–3911
28. Dyer C, Chahuneau V, Smith NA (2013) A simple, fast, and effective reparameterization of IBM Model 2. In: Proceedings of NAACL-HLT, pp 644–648
29. Qi P, Zhang Y, Zhang Y, Bolton J, Manning CD (2020) Stanza: A Python natural language processing toolkit for many human languages. In: Proceedings of the 58th annual meeting of the association for computational linguistics: system demonstrations
30. Ulčar M, Robnik-Šikonja M (2020) High quality ELMo embeddings for seven less-resourced languages. In: Proceedings of the 12th language resources and evaluation conference, LREC 2020, pp 4733–4740
31. Lison P, Tiedemann J (2016) OpenSubtitles2016: extracting large parallel corpora from movie and TV subtitles. In: Proceedings of the 10th international conference on language resources and evaluation LREC
32. Acs J (2014) Pivot-based multilingual dictionary building using Wiktionary. In: Proceedings of the ninth international conference on language resources and evaluation LREC
33. Ljubešić N, Klubička F, Željko Agić, Jazbec IP (2016) New inflectional lexicons and training corpora for improved morphosyntactic annotation of Croatian and Serbian. In: Proceedings of the LREC 2016
34. Tjong Kim Sang EF, De Meulder F (2003) Introduction to the CoNLL-2003 shared task: language-independent named entity recognition. In: Proceedings of CoNLL-2003, pp 142–147
35. Laur S (2013) Nimeüksuste korpus. Center of Estonian Language Resources
36. Ruokolainen T, Kauppinen P, Silfverberg M, Lindén K (2020) A Finnish news corpus for named entity recognition. Lang Resour Eval 54(1):247–272
37. Paikens P, Auziņa I, Garkaje G, Paegle M (2012) Towards named entity annotation of Latvian national library corpus. Front Artif Intell Appl 247:169–175
38. Pinnis M (2012) Latvian and Lithuanian named entity recognition with TildeNER. In: Proceedings of the eighth international conference on language resources and evaluation (LREC'12), pp 1258–1265
39. Krek S, Dobrovoljc K, Erjavec T, Može S, Ledinek N, Holz N, Zupan K, Gantar P, Kuzman T, Čibej J, Arhar Holdt Š, Kavčič T, Škrjanec I, Marko D, Jezeršek L, Zajc A (2019) Training corpus ssj500k 2.2. Slovenian language resource repository CLARIN.SI
40. Klintberg A (2015) Training a Swedish NER-model for Stanford CoreNLP. Accessed 22 July 2021. https://medium.com/@klintcho/training-a-swedish-ner-model-for-stanford-corenlp-part-1-3e3f281a753a
41. Dozat T, Manning CD (2017) Deep biaffine attention for neural dependency parsing. In: Proceedings of 5th international conference on learning representations, ICLR 2017
42. Nivre J, Abrams M, Agić Ž (2020) Universal Dependencies 2.6. http://hdl.handle.net/11234/1-2988
43. Bielinskiene A, Boizou L, Kovalevskaite J (2016) Lithuanian dependency treebank. In: Human language technologies—the baltic perspective: proceedings of the seventh international conference baltic HLT 2016, vol 289, p 107
44. Jurafsky D, Martin JH (2009) Speech and language processing, 2nd edn. Prentice-Hall Inc, New York
45. Steinberger R, Pouliquen B, Hagman J (2002) Cross-lingual document similarity calculation using the multilingual thesaurus Eurovoc. Comput Ling Intell Text Process 101–121
46. Koehn P (2005) Europarl: a parallel corpus for statistical machine translation. In: The tenth machine translation summit proceedings of conference, international association for machine translation, pp 79–86
47. Tiedemann J (2012) Parallel data, tools and interfaces in OPUS. In: Proceedings of the eighth international conference on language resources and evaluation LREC'12
48. Steinberger R, Eisele A, Klocek S, Pilos S, Schlüter P (2012) DGT-TM: A freely available translation memory in 22 languages. In: Proceedings of the 8th international conference on language resources and evaluation LREC 2012
49. Mozetič I, Grčar M, Smailović J (2016) Multilingual Twitter sentiment classification: the role of human annotators. PLOS ONE 11(5)