Abstract

Most of the successful and predominant methods for Bilingual Lexicon Induction (BLI) are mapping-based, where a linear mapping function is learned with the assumption that the word embedding spaces of different languages exhibit similar geometric structures (i.e., approximately isomorphic). However, several recent studies have criticized this simplified assumption showing that it does not hold in general even for closely related languages. In this work, we propose a novel semi-supervised method to learn cross-lingual word embeddings for BLI. Our model is independent of the isomorphic assumption and uses non-linear mapping in the latent space of two independently pre-trained autoencoders. Through extensive experiments on fifteen (15) different language pairs (in both directions) comprising resource-rich and low-resource languages from two different datasets, we demonstrate that our method outperforms existing models by a good margin. Ablation studies show the importance of different model components and the necessity of non-linear mapping.

1 Introduction

In recent years, a plethora of methods have been proposed to learn cross-lingual word embeddings (or CLWE for short) from monolingual word embeddings. Here words with similar meanings in different languages are represented by similar vectors, regardless of their actual language. CLWE enable us to compare the meaning of words across languages, which is key to most multi-lingual applications such as bilingual lexicon induction (Heyman et al., 2017), machine translation (Lample et al., 2018; Artetxe et al., 2018c), or multi-lingual information retrieval (Vulić and Moens, 2015). They also play a crucial role in cross-lingual knowledge transfer between languages (e.g., from resource-rich to low-resource languages) by providing a common representation space (Ruder et al., 2019).

Mikolov et al. (2013a), in their pioneering work, learn a linear mapping function to transform the source embedding space to the target language by minimizing the squared Euclidean distance between the translation pairs of a seed dictionary. They assume that the similarity of geometric arrangements in the embedding spaces is the key reason for their method to succeed as they found linear mapping superior to non-linear mappings with multi-layer neural networks. Subsequent studies propose to improve the model by normalizing the embeddings, imposing an orthogonality constraint on the linear mapper, modifying the objective function, and reducing the seed dictionary size (Artetxe et al., 2016, 2017, 2018a; Smith et al., 2017).

A more recent line of research attempts to eliminate the seed dictionary totally and learn the mapping in a purely unsupervised way (Barone, 2016; Zhang et al., 2017; Conneau et al., 2018; Artetxe et al., 2018b; Xu et al., 2018; Hoshen and Wolf, 2018; Alvarez-Melis and Jaakkola, 2018; Mohiuddin and Joty, 2019, 2020). While not requiring any cross-lingual supervision makes these methods attractive, Vulić et al. (2019) recently show that even the most robust unsupervised method (Artetxe et al., 2018b) fails for a large number of language pairs. They suggest to rethink the main motivations behind fully unsupervised methods showing that with a small seed dictionary (500-1K pairs) their semi-supervised method always outperforms the unsupervised method and does not fail for any language pair. Other concurrent work (Ormazabal et al., 2019; Doval et al., 2019) also advocates for weak supervision in CLWE methods.

Almost all mapping-based CLWE methods, supervised and unsupervised alike, solve the Procrustes problem in the final step or during self-learning (Ruder et al., 2019). This restricts the transformation to be orthogonal linear mappings.
However, learning an orthogonal linear mapping inherently assumes that the embedding spaces of different languages exhibit similar geometric structures (i.e., approximately isomorphic). Several recent studies have questioned this strong assumption and empirically showed that the isomorphic assumption does not hold in general even for two closely related languages like English and German (Søgaard et al., 2018; Patra et al., 2019).

In this work, we propose LNM (Latent space Non-linear Mapping), a novel semi-supervised approach that uses non-linear mapping in the latent space to learn CLWE. It uses minimal supervision from a seed dictionary, while leveraging semantic information from the monolingual word embeddings. As shown in Figure 1, LNM comprises two autoencoders, one for each language. The auto-encoders are first trained independently in a self-supervised way to induce the latent code space of the respective languages. Then, we use a small seed dictionary to learn the non-linear mappings between the two code spaces. To guide our mapping in the latent space, we include two additional constraints: back-translation and original embedding reconstruction. Crucially, our method does not enforce any strong prior constraints like the orthogonality (or isomorphic), rather it gives the model the flexibility to induce the required latent structures such that it is easier for the non-linear mappers to align them in the code space.

In order to demonstrate the effectiveness and robustness of LNM, we conduct extensive experiments on bilingual lexicon induction (BLI) with fifteen (15) different language pairs (in both directions) comprising high- and low-resource languages from two different datasets for different sizes of the seed dictionary. Our results show significant improvements for LNM over the state-of-the-art in most of the tested scenarios. It is particularly very effective for low-resource languages; for example, using 1K seed dictionary, LNM yields about 18% absolute improvements on average over a state-of-the-art supervised method (Joulin et al., 2018). It also outperforms the most robust unsupervised system of Artetxe et al. (2018b) in most of the translation tasks. Interestingly, for resource-rich language pairs, linear autoencoder performs better than non-linear ones. Our ablation study reveals the collaborative nature of LNM’s different components and efficacy of its non-linear mappings in the code space. We open-source our framework at https://ntunlpsg.github.io/project/lnmap/.

2 Background

Limitations of Isomorphic Assumption. Almost all CLWE methods inherently assume that embedding spaces of different languages are approximately isomorphic (i.e., similar in geometric structure). However, recently researchers have questioned this simplified assumption and attributed the performance degradation of existing CLWE methods to the strong mismatches in embedding spaces caused by the linguistic and domain divergences (Søgaard et al., 2019; Ormazabal et al., 2019). Søgaard et al. (2018) empirically show that even closely related languages are far from being isomorphic. Nakashole and Flauger (2018) argue that mapping between embedding spaces of different languages can be approximately linear only at small local regions, but must be non-linear globally. Patra et al. (2019) also recently show that etymologically distant language pairs cannot be aligned properly using orthogonal transformations.

Towards Semi-supervised Methods. A number of recent studies have questioned the robustness of existing unsupervised CLWE methods (Ruder et al., 2019). Vulić et al. (2019) show that even the most robust unsupervised method (Artetxe et al., 2018b) fails for a large number of language pairs; it gives zero (or near zero) BLI performance for 87 out of 210 language pairs. With a seed dictionary of only 500 - 1000 word pairs, their supervised method outperforms unsupervised methods by a wide margin in most language pairs. Other recent work also suggested using semi-supervised methods (Patra et al., 2019; Ormazabal et al., 2019).

Mapping in Latent Space. Mohiuddin and Joty (2019) propose adversarial autoencoder for unsupervised word translation. They use linear autoencoders in their model, and the mappers are also linear. They emphasize the benefit of using latent space over the original embedding space. Although their method is more robust than other existing adversarial models, still it suffers from training instability for distant language pairs.

Our Contributions. Our proposed LNM is independent of the isomorphic assumption. It uses weak supervision from a small seed dictionary, while leveraging rich structural information from monolingual embeddings. Unlike Mohiuddin and Joty (2019), the autoencoders in LNM are
not limited to only linearity. More importantly, it uses non-linear mappers. These two factors contribute to its robust performance even for very low-resource languages (§5). To the best of our knowledge, we are the first to showcase such robust and improved performance with non-linear methods.\footnote{Our experiments with (unsupervised) adversarial training showed very unstable results with the non-linear mappers.}

3 LNMAP Semi-supervised Framework

Let $\mathcal{V}_x = \{v_{x_1}, ..., v_{x_{n_x}}\}$ and $\mathcal{V}_y = \{v_{y_1}, ..., v_{y_{n_y}}\}$ be two sets of vocabulary consisting of $n_x$ and $n_y$ words for a source ($\mathcal{V}_x$) and a target ($\mathcal{V}_y$) language, respectively. Each word $v_{x_i}$ (resp. $v_{y_j}$) has an embedding $x_i \in \mathbb{R}^d$ (resp. $y_j \in \mathbb{R}^d$), trained with any word embedding models, e.g., FastText (Bojanowski et al., 2017). Let $E_{\ell_x} \in \mathbb{R}^{n_x \times d}$ and $E_{\ell_y} \in \mathbb{R}^{n_y \times d}$ be the word embedding matrices for the source and target languages, respectively. We are also given with a seed dictionary $\mathcal{D} = \{(x_1, y_1), ..., (x_k, y_k)\}$ with $k$ word pairs. Our objective is to learn a transformation function $M$ such that for any $v_{x_i} \in \mathcal{V}_x$, $M(x_i)$ corresponds to its translation $y_j$, where $v_{y_j} \in \mathcal{V}_y$. Our approach LNMAP (Figure 1) follows two sequential steps:

(i) Unsupervised latent space induction using monolingual autoencoders (§3.1), and

(ii) Supervised non-linear transformation learning with back-translation and source embedding reconstruction constraints (§3.2).

3.1 Unsupervised Latent Space Induction

We use two autoencoders, one for each language. Each autoencoder comprises an encoder $E_{\ell_x}$ (resp. $E_{\ell_y}$) and a decoder $D_{\ell_x}$ (resp. $D_{\ell_y}$). Unless otherwise stated, the autoencoders are non-linear, where each of the encoder and decoder is a three-layer feed-forward neural network with two non-linear hidden layers. More formally, the encoding-decoding operations of the source autoencoder (autoenc)$_{\ell_x}$ are defined as:

\begin{align*}
    h_{1}^{E_{\ell_x}} &= \phi(\theta_{1}^{E_{\ell_x}} x_i) \quad (1) \\
    h_{1}^{D_{\ell_x}} &= \phi(\theta_{3}^{D_{\ell_x}} z_{x_i}) \quad (4) \\
    h_{2}^{E_{\ell_x}} &= \phi(\theta_{2}^{E_{\ell_x}} h_{1}^{E_{\ell_x}}) \quad (2) \\
    h_{2}^{D_{\ell_x}} &= \phi(\theta_{2}^{D_{\ell_x}} h_{1}^{D_{\ell_x}}) \quad (5) \\
    z_{x_i} &= \theta_{3}^{D_{\ell_x}} h_{2}^{D_{\ell_x}} \quad (3) \\
    \hat{x}_i &= \phi(\theta_{1}^{D_{\ell_x}} h_{2}^{D_{\ell_x}}) \quad (6)
\end{align*}

where $\theta_{1}^{E_{\ell_x}} \in \mathbb{R}^{d \times d}$ and $\theta_{1}^{D_{\ell_x}} \in \mathbb{R}^{d \times d}$ are the parameters of the layers in the encoder and decoder respectively, and $\phi$ is a non-linear activation function; we use Parametric Rectified Linear Unit (PReLU) in all the hidden layers and tanh in the final layer of the decoder (Eq. 6). We use linear activations in the output layer of the encoder (Eq. 3). We train autoenc$_{\ell_x}$ with $l_2$ reconstruction loss as:

\begin{align*}
    L_{\text{autoenc}_{\ell_x}}(\Theta_{E_{\ell_x}}, \Theta_{D_{\ell_x}}) = \frac{1}{n_{x}} \sum_{i=1}^{n_{x}} \|x_i - \hat{x}_i\|^2 \quad (7)
\end{align*}

where $\Theta_{E_{\ell_x}} = \{\theta_{1}^{E_{\ell_x}}, \theta_{2}^{E_{\ell_x}}, \theta_{3}^{E_{\ell_x}}\}$ and $\Theta_{D_{\ell_x}} = \{\theta_{1}^{D_{\ell_x}}, \theta_{2}^{D_{\ell_x}}, \theta_{3}^{D_{\ell_x}}\}$ are the parameters of the encoder and the decoder of autoenc$_{\ell_x}$.

The encoder, decoder and the reconstruction loss for the target autoencoder (autoenc)$_{\ell_y}$ are similarly defined.

3.2 Supervised Non-linear Transformation

Let $q(z_x | x)$ and $q(z_y | y)$ be the distributions of latent codes in autoenc$_{\ell_x}$ and autoenc$_{\ell_y}$, respectively. We have two non-linear mappers: $M$ that translates a source code into a target code, and $N$ that translates a target code into a source code (Figure 1). Both mappers are implemented as a feed-forward neural network with a single hidden layer and tanh activations, and they are trained using the provided seed dictionary $\mathcal{D}$.

Non-linear Mapping Loss. Let $\Theta_{M}$ and $\Theta_{N}$ denote the parameters of the two mappers $M$ and $N$, respectively. While mapping from $q(z_x | x)$ to $q(z_y | y)$, we jointly train the mapper $M$ and the source encoder $E_{\ell_x}$ with the following $l_2$ loss.
\[ \mathcal{L}_{\text{MAP}}(\Theta_{\mathcal{M}}, \Theta_{\text{Enc}_x}) = \frac{1}{k} \sum_{i=1}^{k} \| z_{y_i} - \mathcal{M}(z_{x_i}) \|^2 \]  

The mapping loss for \( \mathcal{N} \) and \( \text{Enc}_y \) is similarity defined. To learn a better transformation function, we enforce two additional constraints to our objective -- back-translation and reconstruction.

**Back-Translation Loss.** To ensure that a source code \( z_{x_i} \in q(z_x|x) \) translated to the target language latent space \( q(z_y|y) \), and then translated back to the original latent space remain unchanged, we enforce the back-translation constraint, that is, \( z_{x_i} \rightarrow \mathcal{M}(z_{x_i}) \rightarrow \mathcal{N}(\mathcal{M}(z_{x_i})) \approx z_{x_i} \). The back-translation (BT) loss from \( q(z_y|y) \) to \( q(z_x|x) \) is

\[ \mathcal{L}_{\text{BT}}(\Theta_{\mathcal{M}}, \Theta_{\mathcal{N}}) = \frac{1}{k} \sum_{i=1}^{k} \| z_{x_i} - \mathcal{N}(\mathcal{M}(z_{x_i})) \|^2 \]  

The BT loss in the other direction (\( z_{y_j} \rightarrow \mathcal{N}(z_{y_j}) \rightarrow \mathcal{M}(\mathcal{N}(z_{y_j})) \approx z_{y_j} \)) is similarly defined.

**Reconstruction Loss.** In addition to back-translation, we include another constraint to guide the mapping further. In particular, we ask the decoder \( D_{\ell_y} \) of \( \text{autoenc}_{\ell_y} \) to reconstruct the original embedding \( x_i \) from the back-translated code \( \mathcal{N}(\mathcal{M}(z_{x_i})) \). We compute this original embedding reconstruction loss for \( \text{autoenc}_{\ell_x} \) as:

\[ \mathcal{L}_{\text{REC}}(\Theta_{\text{Enc}_x}, \Theta_{\text{Dec}_x}, \Theta_{\mathcal{M}}, \Theta_{\mathcal{N}}) = \frac{1}{k} \sum_{i=1}^{k} \| x_i - D_{\ell_y}(\mathcal{N}(\mathcal{M}(z_{x_i}))) \|^2 \]  

The reconstruction loss for \( \text{autoenc}_{\ell_y} \) is defined similarly. Both back-translation and reconstruction lead to more stable training in our experiments. In our ablation study (§5.4), we empirically show the efficacy of the addition of these two constraints.

**Total Loss.** The total loss for mapping a batch of word embeddings from source to target is:

\[ \mathcal{L}_{\ell_x \rightarrow \ell_y} = \mathcal{L}_{\text{MAP}} + \lambda_1 \mathcal{L}_{\text{BT}} + \lambda_2 \mathcal{L}_{\text{REC}} \]  

where \( \lambda_1 \) and \( \lambda_2 \) control the relative importance of the loss components. Similarly we define the total loss for mapping in the opposite direction \( \mathcal{L}_{\ell_y \rightarrow \ell_x} \).

**Remark.** Note that our approach is fundamentally different from existing methods in two ways. First, most of the existing methods directly map the distribution of the source embeddings \( p(x) \) to the distribution of the target embeddings \( p(y) \). Second, they learn a linear mapping function assuming that the two languages’ embedding spaces are nearly isomorphic, which does not hold in general (Søgaard et al., 2018; Patra et al., 2019).

Mapping the representations in the code space using non-linear transformations gives our model the flexibility to induce the required semantic structures in its latent space that could potentially yield more accurate cross-lingual mappings (§5).

### 3.3 Training Procedure

We present the training method of LNMAP in Algorithm 1. In the first step, we pre-train \( \text{autoenc}_{\ell_x} \) and \( \text{autoenc}_{\ell_y} \) separately on the respective monolingual word embeddings. In this unsupervised step, we use the first 200K embeddings. This pre-training induce word semantics (and relations) in the code space (Mohiuddin and Joty, 2019).

The next step is the self-training process, where we train the mappings along with the autoencoders using the seed dictionary in an iterative manner. We keep a copy of the original dictionary \( D \); let us call it \( D_{\text{orig}} \). We first update the mapper \( \mathcal{M} \) and the source encoder \( \text{Enc}_x \) on the mapping loss (Eq. 8). The mappers (both \( \mathcal{M} \) and \( \mathcal{N} \)) then go through two more updates, one for back-translation (Eq. 9) and the other for reconstruction of the source embedding (Eq. 10). The entire source autoencoder \( \text{autoenc}_{\ell_x} \) (both \( \text{Enc}_x \) and \( D_{\ell_x} \)) in this stage gets updated only on the reconstruction loss.

After each iteration of training (step \( i \) in Alg. 1), we induce a new dictionary \( D_{\text{new}} \) using the learned encoders and mappers. To find the nearest target word \( (y_j) \) of a source word \( (x_i) \) in the target latent space, we use the Cross-domain Similarity Local Scaling (CSLS) measure which works better than simple cosine similarity in mitigating the hubness problem (Conneau et al., 2018). It penalizes the words that are close to many other words in the target latent space. To induce the dictionary, we compute CSLS for \( K \) most frequent source and target words and select the translation pairs that are nearest neighbors of each other according to CSLS.

For the next iteration of training, we construct the dictionary \( D \) by merging \( D_{\text{orig}} \) with the \( l \) most similar (based on CSLS) word pairs from \( D_{\text{new}} \).
Algorithm 1: Training LNM

**Input**: Word embedding matrices: $\mathcal{E}_x, \mathcal{E}_y$, seed dictionary: $\mathcal{D}$, and increment count $C$

// Unsup. latent space induction
1. Train $\text{autoenc}_x$ and $\text{autoenc}_y$ separately for some epochs on monolingual word embeddings

// Sup. non-linear transformation
2. $\text{iter} = 0; \mathcal{D}_{\text{orig}} = \mathcal{D}$
3. do
   i. for $n_{\text{epochs}}$ do
      (a) Sample a mini-batch from $\mathcal{D}$
      (b) Update mapper $\mathcal{M}$ and $\mathcal{E}_y$ on the non-linear mapping loss
      (c) Update mappers $\mathcal{M}$ and $\mathcal{N}$ on the back-translation loss
      (d) Update mappers ($\mathcal{M}, \mathcal{N}$) and $\text{autoenc}_x$ on the reconstruction loss
   end
   ii. Induce a new dictionary $\mathcal{D}_{\text{new}}$ of size: $\text{iter} \times C$
   iii. Create a new dictionary, $\mathcal{D} = \mathcal{D}_{\text{orig}} \cup \mathcal{D}_{\text{new}}$
3. end

while not converge;

We set $l$ as $l = \text{iter} \times C$, where $\text{iter}$ is the current iteration number and $C$ is a hyperparameter. This means we incrementally update the dictionary size. This is because the induced dictionary at the initial iterations is likely to be noisy. As the training progresses, the model becomes more mature, and the induced dictionary pairs become better. For convergence, we use the criterion: if the difference between the average similarity scores of two successive iteration steps is less than a threshold (we use $1 \times 10^{-6}$), then stop the training process.

4 Experimental Settings

We evaluate our approach on bilingual lexicon induction, also known as word translation.

4.1 Datasets

To demonstrate the effectiveness of our method, we evaluate our models against baselines on two popularly used datasets: MUSE (Conneau et al., 2018) and VecMap (Dinu et al., 2015).

The MUSE dataset consists of FastText monolingual embeddings of 300 dimensions (Bojanowski et al., 2017) trained on Wikipedia monolingual corpus and gold dictionaries for 110 language pairs.\(^2\) To show the generality of different methods, we consider 15 different language pairs with $15 \times 2 = 30$ different translation tasks encompassing resource-rich and low-resource languages from different language families. In particular, we evaluate on English (En) from/to Spanish (Es), German (De), Italian (It), Russian (Ru), Arabic (Ar), Malay (Ms), Finnish (Fi), Estonian (Et), Turkish (Tr), Greek (El), Persian (Fa), Hebrew (He), Tamil (Ta), Bengali (Bn), and Hindi (Hi). We differentiate between high- and low-resource languages by the availability of NLP-resources in general.

The VecMap dataset (Dinu et al., 2015; Artetxe et al., 2018a) is a more challenging dataset and contains monolingual embeddings for English, Spanish, German, Italian, and Finnish.\(^3\) According to Artetxe et al. (2018b), existing unsupervised methods often fail to produce meaningful results on this dataset. English, Italian, and German embeddings were trained on WacKy crawling corpora using CBOW (Mikolov et al., 2013b), while Spanish and Finnish embeddings were trained on WMT News Crawl and Common Crawl, respectively.

4.2 Baseline Methods

We compare our proposed LNM with several existing methods comprising supervised, semi-supervised, and unsupervised models. For each baseline model, we conduct experiments with the publicly available code. In the following, we give a brief description of the baseline models.

**Supervised & Semi-supervised Methods.**

(a) Artetxe et al. (2017) propose a self-learning framework that performs two steps iteratively until

\[^{2}\text{https://github.com/facebookresearch/MUSE}\]

\[^{3}\text{https://github.com/artetxem/vecmap/}\]
convergence. In the first step, they use the dictionary (starting with the seed dictionary) to learn a linear mapping, which is then used in the second step to induce a new dictionary.

(b) Artetxe et al. (2018a) propose a multi-step framework that generalizes previous studies. Their framework consists of several steps: whitening, orthogonal mapping, re-weighting, de-whitening, and dimensionality reduction.

(c) Conneau et al. (2018) compare their unsupervised model with a supervised baseline that learns an orthogonal mapping between the embedding spaces by iterative Procrustes refinement. They also propose CSLS for nearest neighbour search.

(d) Joulin et al. (2018) show that minimizing a convex relaxation of the CSLS loss significantly improves the quality of bilingual word vector alignment. Their method achieves state-of-the-art results for many languages (Patra et al., 2019).

(e) Jawanpuria et al. (2019) propose a geometric approach where they decouple CLWE learning into two steps: (i) learning rotations for language-specific embeddings to align them to a common space, and (ii) learning a similarity metric in the common space to model similarities between the embeddings of the two languages.

(f) Patra et al. (2019) propose a semi-supervised technique that relaxes the isomorphic assumption while leveraging both seed dictionary pairs and a larger set of unaligned word embeddings.

Unsupervised Methods.

(a) Conneau et al. (2018) are the first to show impressive results for unsupervised word translation by pairing adversarial training with effective refinement methods. Given two monolingual word embeddings, their adversarial training plays a two-player game, where a linear mapper (generator) plays against a discriminator. They also impose the orthogonality constraint on the mapper. After adversarial training, they use the iterative Procrustes solution similar to their supervised approach.

(b) Artetxe et al. (2018b) learn an initial dictionary by exploiting the structural similarity of the embeddings in an unsupervised way. They propose a robust self-learning to improve it iteratively. This model is by far the most robust and best performing unsupervised model (Vulić et al., 2019).

(c) Mohiuddin and Joty (2019) use adversarial autoencoder for unsupervised word translation.

They use linear autoencoders in their model, and the mappers are also linear.

4.3 Model Variants and Settings

We experiment with two variants of our model: the default LNMAP that uses non-linear autoencoders and LNMAP (LIN. AE) that uses linear autoencoders. In both the variants, the mappers are non-linear. We train our models using stochastic gradient descent (SGD) with a batch size of 128, a learning rate of $10^{-4}$, and a step learning rate decay schedule. During the dictionary induction process in each iteration, we consider $K = 15000$ most frequent words from the source and target languages. For dictionary update, we set $C = 2000$.

5 Results and Analysis

We present our results on low-resource and resource-rich languages from MUSE dataset in Tables 1 and 2, respectively, and the results on VecMap dataset in Table 3. We present the results in precision@1, which means how many times one of the correct translations of a source word is predicted as the top choice. For each of the cases, we show results on seed dictionary of three different sizes including 1-to-1 and 1-to-many mappings; “1K Unique” and “5K Unique” contain 1-to-1 mappings of 1000 and 5000 source-target pairs respectively, while “5K All” contains 1-to-many mappings of all 5000 source and target words, that is, for each source word there can be multiple target words. Through experiments and analysis, our goal is to assess the following questions.

(i) Does LNMAP improve over the best existing methods in terms of mapping accuracy on low-resource languages (§5.1)?

(ii) How well does LNMAP perform on resource-rich languages (§5.2)?

(iii) What is the effect of non-linearity in the autoencoders? (§5.3)

(iv) Which components of LNMAP attribute to improvements (§5.4)?

5.1 Performance on Low-resource Languages

Most of the unsupervised models fail in the majority of the low-resource languages (Vulić et al., 2019). On the other hand, the performance of supervised models on low-resource languages was not satisfactory, especially with small seed dictionary.
Hence, we first compare LNMAP’s performance on these languages. From Table 1, we see that on average LNMAP outperforms every baseline by a good margin (1.1% - 5.2% from the best baselines).

For “1K Unique” dictionary, LNMAP exhibits impressive performance. In all the 20 translation tasks, it outperforms all the (semi-)supervised baselines by a wide margin. If we compare with Joulin et al. (2018), a state-of-the-art supervised model, LNMAP’s average improvement is ~18%, which is remarkable. Compared to other baselines, the average margin of improvement is also quite high – 9.9%, 14.6%, 5.2%, 9.7%, and 8.0% gains over Artetxe et al. (2017), Artetxe et al. (2018a), Conneau et al. (2018), Jawanpuria et al. (2019), and Patra et al. (2019), respectively. We see that among the supervised baselines, Conneau et al. (2018)’s model performs better than others.

If we increase the dictionary size, we can still see the dominance of LNMAP over the baselines. For “5K Only” seed dictionary, it performs better than the baselines on 14/20 translation tasks, while for “5K All” seed dictionary, the best performance by LNMAP is on 13/20 translation tasks.

One interesting thing to observe is that, under resource-constrained setup LNMAP’s performance is impressive, making it suitable for very low-resource languages like En-Ta, En-Bn, and En-Hi.

Now if we look at the performance of unsupervised baselines on low-resource languages, we see that Conneau et al. (2018)’s model fails to converge on the majority of the translation tasks (12/20), while the model of Mohiuddin and Joty (2019) fails to converge on En↔Ta, En↔Bn, and En↔Hi. Although the most robust unsupervised method of Artetxe et al. (2018b) performs better than the other unsupervised approaches, it still fails to converge on En↔Ta tasks. If we compare its performance with LNMAP, we see that our model outperforms the best unsupervised model of Artetxe et al. (2018b) on 18/20 low-resource translation tasks.

### 5.2 Results on Resource-rich Languages

Table 2 shows the results for 5 resource-rich language pairs (10 translation tasks) from the MUSE dataset. We notice that our model achieves the highest accuracy in all the tasks for “1K Unique”, 4 tasks for “5K Unique”, 3 for “5K All”.

We show the results on the VecMap dataset in Table 3, where there are 3 resource-language pairs.
pairs, and one low-resource pair (En-Fi) with a total of 8 translation tasks. Overall, we have similar observations as in MUSE – our model outperforms other models on 7 tasks for “1K Unique”, 4 tasks for “5K Unique”, and 4 for “5K All”.

### 5.3 Effect of Non-linearity in Autoencoders

The comparative results between our model variants in Tables 1 - 3 reveal that LNMAP (with non-linear autoencoders) works better for low-resource languages, whereas LNMAP (Lin. AE) works better for resource-rich languages. This can be explained by the geometric similarity between the embedding spaces of the two languages.

In particular, we measure the geometric similarity of the language pairs using the Gromov-Hausdorff (GH) distance (Patra et al., 2019), which is recently proposed to quantitatively estimate isometry between two embedding spaces.\(^4\)

From the measurements (Tables 1-2), we see that etymologically close language pairs have lower GH distance compared to etymologically distant and low-resource language pairs.\(^5\) Low-resource language pairs’ high GH distance measure implies that English and those languages embedding spaces are far from isomorphism. Hence, we need strong non-linearity for those distant languages.

### 5.4 Dissecting LNMAP

We further analyze our model by dissecting it and measuring the contribution of its different components. Specifically, our goal is to assess the contribution of back-translation, reconstruction, non-linearity in the mapper, and non-linearity in the autoencoder. We present the ablation results in Table 4 on 8 translation tasks from 4 language pairs consisting of 2 resource-rich and 2 low-resource languages. We use MUSE dataset for this purpose.

\(^4\)https://github.com/joelmoniz/BLISS

\(^5\)We could not compute GH distances for the VecMap dataset; the metric gives ‘inf’ in the BLISS framework.
Table 3: Word translation accuracy (P@1) on VecMap dataset using CBOW embeddings.

<table>
<thead>
<tr>
<th>Resource-rich</th>
<th>Low-Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>En-Es</td>
</tr>
<tr>
<td>LNMAP</td>
<td>80.1</td>
</tr>
<tr>
<td>Recon. loss</td>
<td>79.4</td>
</tr>
<tr>
<td>Back-trans.</td>
<td>79.8</td>
</tr>
<tr>
<td>Linear mapper</td>
<td>78.8</td>
</tr>
<tr>
<td>Procrustes sol.</td>
<td>75.9</td>
</tr>
<tr>
<td>Linear autoenc.</td>
<td>83.5</td>
</tr>
</tbody>
</table>

Table 4: Ablation study of LNMAP with “1K Unique” dictionary. ⊗ indicates the component is removed from the full model, and “⊕” indicates the component is added by replacing the corresponding component.

6 Conclusions

We have presented a novel semi-supervised framework LNMAP to learn the cross-lingual mapping between two monolingual word embeddings. Apart from exploiting weak supervision from a small (1K) seed dictionary, our LNMAP leverages the information from monolingual word embeddings. In contrast to the existing methods that directly map word embeddings using the isomorphic assumption, our framework is independent of any such strong prior assumptions. LNMAP first learns to transform the embeddings into a latent space and then uses a non-linear transformation to learn the mapping. To guide the non-linear mapping further, we include constraints for back-translation and original embedding reconstruction.

Extensive experiments with fifteen different language pairs comprising high- and low-resource languages show the efficacy of non-linear transformations, especially for low-resource and distant languages. Comparison with existing supervised, semi-supervised, and unsupervised baselines show that LNMAP learns a better mapping. With an in-depth ablation study, we show that different components of LNMAP works in a collaborative nature.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments. Shafiq Joty would like to thank the funding support from NRF (NRF2016IDM-TRANS001-062), Singapore.
References


Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018b. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In ACL.


A.1 Reproducibility Settings
- Computing infrastructure - Linux machine with a single GTX 1080 Ti GPU
- PyTorch version 1.2.0
- CUDA version 10.0
- cuDNN version 7.6.0
- Average runtime - 15-20 minutes

A.2 Optimal Hyperparameters

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoder #layers</td>
<td>3</td>
</tr>
<tr>
<td>input dim</td>
<td>300</td>
</tr>
<tr>
<td>hidden dim</td>
<td>350-400</td>
</tr>
<tr>
<td>output dim</td>
<td>350-400</td>
</tr>
<tr>
<td>hidden non-linearity</td>
<td>PReLU</td>
</tr>
<tr>
<td>output non-linearity</td>
<td>linear</td>
</tr>
<tr>
<td>Decoder #layers</td>
<td>3</td>
</tr>
<tr>
<td>input dim</td>
<td>350-400</td>
</tr>
<tr>
<td>hidden dim</td>
<td>350-400</td>
</tr>
<tr>
<td>output dim</td>
<td>300</td>
</tr>
<tr>
<td>hidden non-linearity</td>
<td>PReLU</td>
</tr>
<tr>
<td>output non-linearity</td>
<td>tanh</td>
</tr>
</tbody>
</table>

Table 5: Optimal hyper-parameter settings for autoencoder.
<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>linear/non-linear</td>
</tr>
<tr>
<td>#layers</td>
<td>2</td>
</tr>
<tr>
<td>input dim</td>
<td>350-400</td>
</tr>
<tr>
<td>hidden dim</td>
<td>400</td>
</tr>
<tr>
<td>output dim</td>
<td>350-400</td>
</tr>
<tr>
<td>hidden non-linearity</td>
<td>tanh</td>
</tr>
<tr>
<td>output non-linearity</td>
<td>linear</td>
</tr>
</tbody>
</table>

Table 6: Optimal hyper-parameter settings for mapper.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>normalization</td>
<td>renorm, center, renorm</td>
</tr>
<tr>
<td>#iterations</td>
<td>dynamic</td>
</tr>
<tr>
<td>sup. dict size</td>
<td>1K-5K</td>
</tr>
<tr>
<td>batch size</td>
<td>128</td>
</tr>
<tr>
<td>autoenc. epochs</td>
<td>25</td>
</tr>
<tr>
<td>mapper epochs</td>
<td>100</td>
</tr>
<tr>
<td>nearest-neighbor</td>
<td>CSLS</td>
</tr>
<tr>
<td>autoenc. optimizer</td>
<td>SGD</td>
</tr>
<tr>
<td>autoenc. learning-rate</td>
<td>0.0001</td>
</tr>
<tr>
<td>mapper optimizer</td>
<td>SGD</td>
</tr>
<tr>
<td>mapper learning-rate</td>
<td>0.0001</td>
</tr>
<tr>
<td>mapping-loss weight</td>
<td>1.0</td>
</tr>
<tr>
<td>cycle-loss weight</td>
<td>1.0</td>
</tr>
<tr>
<td>recons.-loss weight</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Table 7: Optimal hyper-parameter settings for LN-MAP training.