Unsupervised Word Translation with Adversarial Autoencoder

Tasnim Mohiuddin
Nanyang Technological University
School of Computer Science
and Engineering
mohi0004@e.ntu.edu.sg

Shafiq Joty
Nanyang Technological University
School of Computer Science and
Engineering
Salesforce Research Asia
srjoty@ntu.edu.sg

Crosslingual word embeddings learned from monolingual embeddings have a crucial role in many downstream tasks, ranging from machine translation to transfer learning. Adversarial training has shown impressive success in learning crosslingual embeddings and the associated word translation task without any parallel data by mapping monolingual embeddings to a shared space. However, recent work has shown superior performance for non-adversarial methods in more challenging language pairs. In this article, we investigate adversarial autoencoder for unsupervised word translation and propose two novel extensions to it that yield more stable training and improved results. Our method includes regularization terms to enforce cycle consistency and input reconstruction, and puts the target encoders as an adversary against the corresponding discriminator. We use two types of refinement procedures sequentially after obtaining the trained encoders and mappings from the adversarial training, namely, refinement with Procrustes solution and refinement with symmetric re-weighting. Extensive experimentations with high- and low-resource languages from two different data sets show that our method achieves better performance than existing adversarial and non-adversarial approaches and is also competitive with the supervised system. Along with performing comprehensive ablation studies to understand the contribution of different components of our adversarial model, we also conduct a thorough analysis of the refinement procedures to understand their effects.

1. Introduction

Learning crosslingual word embeddings has been shown to be an effective way to transfer knowledge from one language to another for many key linguistic tasks,

Submission received: 21 March 2019; revised version received: 8 November 2019; accepted for publication: 29 January 2020.

https://doi.org/10.1162/COLI_a_00374

including machine translation, named entity recognition, part-of-speech tagging, and parsing (Ruder, Vulic, and Sogaard 2017). Whereas earlier efforts used large parallel corpora to solve the associated word alignment problem (Luong, Pham, and Manning 2015), broader applicability demands methods to relax this requirement because acquiring a large corpus of parallel data is not feasible in most scenarios. Recent methods instead use embeddings learned from monolingual corpora, and then learn a linear mapping from one language to another with the underlying assumption that two embedding spaces exhibit similar geometric structures, also known as the **isomorphic** assumption. This allows the model to learn effective crosslingual representations without expensive supervision.

Given monolingual word embeddings of two languages, Mikolov, Le, and Sutskever (2013) show that a linear mapping can be learned from a seed dictionary of 5,000 word pairs by minimizing the sum of squared Euclidean distances between the mapped vectors and the target vectors. Subsequent studies (Xing et al. 2015; Artetxe, Labaka, and Agirre 2016, 2017; Smith et al. 2017) propose to improve the model by normalizing the embeddings, imposing an orthogonality constraint on the mapper, and modifying the objective function. These methods assume some supervision in the form of a seed dictionary, although recently fully unsupervised methods have shown competitive results. Zhang et al. (2017a, 2017b) first reported encouraging results for unsupervised models with **adversarial training**. Conneau et al. (2018) improved this approach with post-mapping refinements, showing impressive results for many language pairs. Their learned mapping was then successfully used to train a fully unsupervised neural machine translation system (Lample et al. 2018a, 2018b).

Although successful, adversarial training has been criticized for not being stable and failing to converge, inspiring researchers to propose **non-adversarial** methods more recently (Xu et al. 2018; Hoshen and Wolf 2018; Alvarez-Melis and Jaakkola 2018; Artetxe, Labaka, and Agirre 2018b). In particular, Artetxe, Labaka, and Agirre (2018b) show that the adversarial methods of Conneau et al. (2018) and Zhang et al. (2017a, 2017b) fail for many difficult language pairs.

In this article, we revisit adversarial training and propose a number of key improvements that yield more robust training and improved mappings. Our main idea is to learn the crosslingual mapping in a projected latent space (code space) and add more constraints to guide the unsupervised mapping in this space. We accomplish this by proposing a novel **adversarial autoencoder** framework (Makhzani et al. 2015), where adversarial mapping is done at the latent code space as opposed to the original embedding space. This gives the model the flexibility to automatically induce the required geometric structures in its code space that could potentially yield better mappings. Figure 1 shows a conceptual demonstration of our main idea.

Søgaard, Ruder, and Vulić (2018) recently found that the isomorphic assumption made by most existing methods does not hold in general even for two closely related languages like English and German. In their words, "approaches based on this assumption have important limitations" (page 778). By performing nonlinear transformations of the original embeddings into their respective code spaces in the autoencoders and then mapping the latent codes of two languages through adversarial training, our approach therefore departs from the isomorphic assumption.

In our adversarial training, not only the mapper but also the target encoder is trained to fool the language discriminator. This forces the discriminator to improve its discrimination skills, which in turn pushes the mapper to generate indistinguishable translation. To guide the mapping, we include two additional constraints. Our first

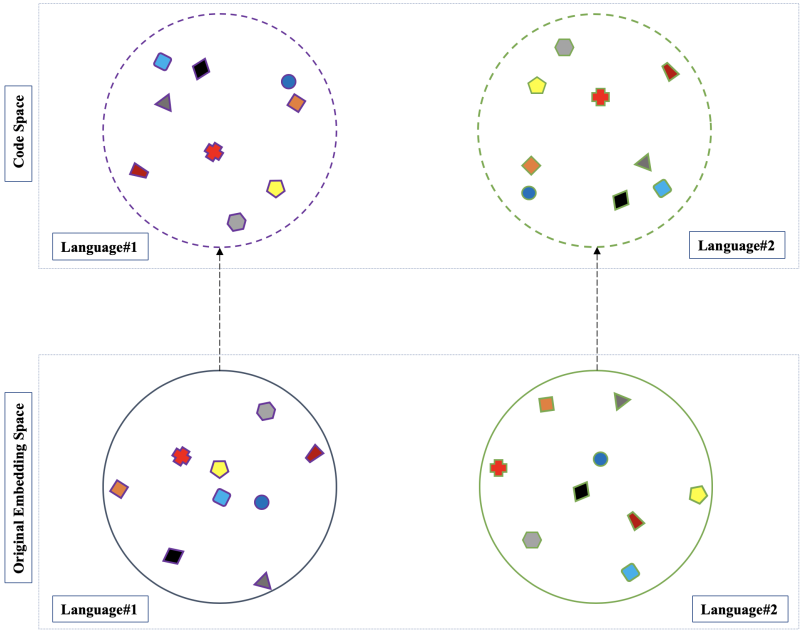


Figure 1
Conceptual demonstration of our proposed crosslingual mapping method. Identical shapes denote the similar meaning words in the two languages. In the original embedding space, the geometric structures of the words in the two languages are different (non-isomorphic). The geometric structures become similar (nearly isomorphic) in the latent code space.

constraint enforces cycle consistency so that code vectors after being translated from one language to another, and then translated back to their source space, remain close to the original vectors. The second constraint ensures reconstruction of the original input word embeddings from the back-translated codes. This grounding step forces the model to retain word semantics during the mapping process and yields more stable training.

The initial bilingual dictionary induced by adversarial training (or any other unsupervised method) is generally of lower quality than what could be achieved by a supervised method. Conneau et al. (2018) and Artetxe, Labaka, and Agirre (2018b) propose fine-tuning methods to refine the initial mappings. In particular, Conneau et al. (2018) refine the initial mapping by iteratively solving the **Procrustes** problem and applying a dictionary induction step. Artetxe, Labaka, and Agirre (2018b) propose a multistep dictionary induction framework. Our work incorporates two types of refinement procedures, namely, refinement with Procrustes solution and refinement with symmetric re-weighting, a step proposed by Artetxe, Labaka, and Agirre (2018b). We perform refinement with the Procrustes solution in the code space, while refinement with symmetric re-weighting is done with the original word embeddings. This way our overall framework combines the two refinement procedures to get the best of both.

In order to demonstrate the effectiveness and robustness of our approach, we conduct a series of experiments with eight different language pairs (in both directions) comprising high- and low-resource languages from two different data sets. We also perform extensive ablation studies to understand the contribution of different components of our

adversarial autoencoder model and different refinement procedures. Our main findings are the following.

(i) Our adversarial method is more robust and yields significant gains over the adversarial method of Conneau et al. (2018) for all translation tasks in all evaluation measures.

(ii) Our method with adversarial autoencoder exhibits better performance than other existing supervised and unsupervised methods in most of the translation tasks.

(iii) The ablation study of our adversarial autoencoder model reveals that cycle consistency contributes the most, while adversarial training of the target encoder and post-cycle reconstruction also have significant effects.

(iv) The in-depth analysis of the refinement procedures shows that symmetric re-weighting is a powerful method and complements the Procrustes solution based refinement method.

We have released our source code at `https://ntunlpsg.github.io/project/unsup-word-translation/`.

In the rest of the article, we first review the related supervised and unsupervised (both adversarial and non-adversarial) word translation models in Section 2, then present our proposed unsupervised approach with adversarial autoencoder and refinement procedures in Section 3. In Section 4, we present the experimental settings—the data sets, and the supervised and the unsupervised baselines that we compare with. We present our results with in-depth analysis in Section 5. Finally, we summarize our contributions with future directions in Section 6.

2. Related Work

In recent years, a number of methods have been proposed to learn a bilingual dictionary from monolingual word embeddings.[1] Many of these methods use an initial seed dictionary. However, more recently, researchers have attempted to eliminate the seed dictionary totally and learn the mapping in a purely unsupervised way. In this section, we give an overview of existing supervised and unsupervised word translation methods. We also discuss the hubness problem that often occurs in these methods and approaches to alleviate the effect of this problem.

2.1 Supervised Models

Mikolov, Le, and Sutskever (2013) first show encouraging results by learning a linear mapping from the source to the target language word embedding space using a seed dictionary of $5,000$ pairs. In their view, the key reason behind the good performance of their model is the similarity of geometric arrangements in vector spaces of the embeddings of different languages. Given a seed dictionary $\mathcal{D} = \{x_i, y_i\}_{i=1}^{n}$, where x_i is the embedding of a word in the source language, and y_i is the embedding of its

1 See Ruder, Vulic, and Sogaard (2017) for a nice survey.

translation in the target language, they learn a linear mapping by solving the following regression (also known as the **Ordinary Least Squares** or **OLS**) problem:

$$W_{\text{OLS}} = \min_{W} \sum_{i=1}^{n} \|Wx_i - y_i\|^2 \tag{1}$$

This equation can be solved by using gradient-based methods such as gradient descent. It also has a closed-form solution: $W_{\text{OLS}} = (X^T X)^{-1} X^T Y$, where X and Y are the matrices containing the embeddings of source and target words in the seed dictionary. For translating a new source word, they map the corresponding word embedding to the target space using the learned mapping W_{OLS} and find the nearest target word. In their approach, they found that simple linear mapping works better than nonlinear mappings with multilayer neural networks.

Xing et al. (2015) identified some inconsistencies between the objective function to learn the embedding and the objective to learn the linear mapping. They solve this by enforcing the word vectors to be of unit length during the learning of the embeddings. Instead of using Euclidean distance in the objective function for learning the mapping, they propose to use **cosine similarity**:

$$W_{\text{COS}} = \max_{W} \sum_{i=1}^{n} (Wx_i)^T y_i \tag{2}$$

To preserve unit length after mapping, they enforce the orthogonality constraint on W, i.e., $WW^T = I$. As a result, the inner product in Equation (2) is equivalent to cosine similarity.

Instead of learning a mapping from the source to the target embedding space, Faruqui and Dyer (2014) use a technique based on Canonical Correlation Analysis to project both source and target embeddings to a common low-dimensional space, where the correlation of the word pairs in the seed dictionary is maximized.

Artetxe, Labaka, and Agirre (2016) show that the above methods are variants of the same core optimization objective and propose a general framework that explains the relation between the methods of Mikolov, Le, and Sutskever (2013), Xing et al. (2015), and Faruqui and Dyer (2014). The orthogonality constraint on W and the unit-length normalization of word embeddings ensure that Equations (1) and (2) are equivalent. Use of mean-centering along each dimension of the word embeddings for maximum expected covariance shows that the method is closely related to the method proposed by Faruqui and Dyer (2014). Artetxe, Labaka, and Agirre (2016) also empirically show the efficacy of orthogonality constraint on W. Under the orthogonality constraint, they show that the optimization problem of Equation (1) has an exact solution:

$$W_{\text{ORT}} = VU^T \tag{3}$$

where $Y^T X = U\Sigma V^T$ is the Singular Value Decomposition (SVD) of $Y^T X$. Smith et al. (2017) show that this analytical solution is closely related to the orthogonal Procrustes solution.

In their follow-up work, Artetxe, Labaka, and Agirre (2017) obtain competitive results using a seed dictionary of only 25 word pairs. They propose a self-learning framework that performs two steps iteratively until convergence. In the first step, they use the dictionary (starting with the seed) to learn a linear mapping, which is then used in the second step to induce a new dictionary.

In their more recent work, Artetxe, Labaka, and Agirre (2018a) propose a *multistep framework* that generalizes previous studies. Their framework consists of several steps: whitening, orthogonal mapping, re-weighting, de-whitening, and dimensionality reduction. They show that existing methods can be explained in terms of these steps. For example, *regression methods* such as the method of Mikolov, Le, and Sutskever (2013) correspond to the case where whitening is applied to both the source and target language embeddings, re-weighting is applied only to source language embeddings, and de-whitening is applied to both language embeddings. The *canonical method* of Faruqui and Dyer (2014) corresponds to the case where whitening is applied to both the source and target language embeddings, and dimensionality reduction is applied to both, but re-weighting and de-whitening are not performed. Similarly, *orthogonal methods* such as the methods of Artetxe, Labaka, and Agirre (2016) and Smith et al. (2017) correspond to the case where only orthogonal mapping is applied.

2.2 Unsupervised Models

As mentioned, a more recent line of research attempts to eliminate the seed dictionary and learn the bilingual mapping in a completely unsupervised way. Initial approaches used adversarial methods, but some non-adversarial methods have also been proposed more recently.

2.2.1 Adversarial Methods. Barone (2016) is the first to propose a model for solving bilingual word translation in an unsupervised way. He initially used an adversarial network similar to Conneau et al. (2018), and found that the mapper (which is also the encoder) translates everything to a single embedding, commonly known as the **mode collapse** issue (Goodfellow 2017). To preserve diversity in mapping, he then used a decoder to reconstruct the source embedding from the mapped embedding, extending the framework to an adversarial autoencoder. His analysis (qualitative) shows promising but not competitive with methods that use bilingual seeds. He suspected issues with adversarial training and with the underlying isomorphic assumption.

In our work, we successfully address these issues with an improved framework that also relaxes the isomorphic assumption. Our framework comprises two separate autoencoders, one for each language, which allows us to put more constraints to guide the mapping through adversarial training. We also distinguish the role of an encoder from the role of a mapper. The encoder projects embeddings to latent code vectors, which are then translated by the mapper.

Zhang et al. (2017a) first show encouraging results on unsupervised word translation with adversarial methods. They propose the following three unsupervised models.

(i) **Unidirectional transformation model**: This model is similar in spirit to the model of Conneau et al. (2018). The generator tries to transform the source embeddings such that they are indistinguishable from the target embeddings, whereas the discriminator tries to distinguish the real target embeddings from the ones that are generated by the generator.

(ii) **Bidirectional transformation model**: There are two generators in this model that transform embeddings from one language space to another language space. Two separate discriminators for each language are used to distinguish the real embeddings from the transformed ones.

(iii) **Adversarial autoencoder model**: In this model, the generator is responsible for transforming source embeddings to target space not only to make them indistinguishable by the discriminator but also for back-translating them to the source space. For this reason, they introduce reconstruction loss. Although they call it an adversarial autoencoder model, their approach is similar to the cycle GAN (Zhu et al. 2017).

To aid training, they incorporate additional techniques like noise injection which works as a regularizer. For selecting the best model, they rely on sharp drops of the discriminator accuracy. In their follow-up work (Zhang et al. 2017b), they minimize Earth-Mover's distance between the distribution of the transformed source embeddings and the distribution of the target embeddings. They propose two models for this: (*i*) **Wasserstein GAN (WGAN)**, which minimizes the Wasserstein distance (closely related to Earth-Mover's distance) between the transformed source distribution and the target distribution, and (*ii*) **EMDOT**, which minimizes the Earth-Mover's distance under orthogonal transformation.

Conneau et al. (2018) is 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 (Goodfellow et al. 2014). The discriminator is trained to distinguish between the original target embedding and the mapped source embedding in the target space. On the other hand, the mapper is jointly trained to fool the discriminator. They also impose the orthogonality constraint on the mapper to preserve the monolingual quality of the embeddings and to make the training more stable. After adversarial training, they extract a synthetic dictionary from the resulting shared embedding space. To ensure a high-quality synthetic dictionary, they consider the most frequent words and retain only mutual nearest neighbors in the dictionary induction process. For fine-tuning the linear mapper, they use the Procrustes solution in Equation (3), which is the closed form solution of Equation (1) under the orthogonality constraint. Similar to Artetxe, Labaka, and Agirre (2017), they do the fine-tuning in an iterative way. Because their method is purely unsupervised, instead of using any bilingual dictionary, they introduce an unsupervised selection metric for selecting the best model. This metric is highly correlated with the mapping quality and quantifies the closeness of the source and target embedding spaces. They use it as a stopping criterion as well as to select the best hyperparameters of the model.

Instead of using an adversarial loss, Xu et al. (2018) use Sinkhorn distance, another distributional similarity measure. They optimize the linear mapping in both directions (source to target and vice versa) for each language pair in the way that the source embeddings mapped to the target space match the distribution of the target embeddings. Moreover, when the mapped source embeddings from the target space are back-translated to the source space, they are maximally close to the original source embeddings, also known as **cycle consistency** (Zhu et al. 2017). To avoid the problem of getting stuck in a poor local minima, their model requires a good initial setting of the parameters. To ensure this, in the first phase of the training, they optimize the Wasserstein distance instead of Sinkhorn distance.

Similar to Zhang et al. (2017a) and (Xu et al. 2018), we also incorporate cycle consistency along with the adversarial loss to train our adversarial autoencoder. However, whereas all the existing methods learn the mapping in the original embedding space, our approach learns it in the latent code space, considering both the mapper and the

target encoder as adversarial with the discriminator. In addition, we use a post-cycle reconstruction to guide the mapping.

2.2.2 Non-Adversarial Methods. Despite being successful, adversarial models have been criticized for instability and failing to converge, prompting researchers to investigate non-adversarial methods for unsupervised mapping of embeddings, as we summarize them here.

Artetxe, Labaka, and Agirre (2018b) learn an initial dictionary by exploiting the structural similarity of the embeddings and use a robust self-learning algorithm to improve it iteratively. Their proposed method consists of the following four sequential steps.

(i) **Preprocessing**: They length-normalize the embeddings, mean-center along each dimension, and then length-normalize them again to ensure the unit length of the embeddings.

(ii) **Fully unsupervised initialization**: To create an initial dictionary, they follow the observation that in the similarity matrix of all words, each word has a different distribution of similarity values and equivalent words in two different languages have a similar "similarity" distribution. So under the strict isometric assumption, sorted similarity vectors of two equivalent words from two different languages will be the same. Based on this assumption, they induce an initial dictionary. In practice, the isometric assumption does not hold (Søgaard, Ruder, and Vulić 2018), thus resulting in a noisy dictionary.

(iii) **Robust self-learning**: This step iteratively improves the initial solution. Their goal is to learn two linear mappers to map the source and the target embeddings to a shared embedding space. Xie et al. (2018) show that under orthogonality constraint, mapping from the source to the target space is equivalent to mapping both source and target to a shared embedding space. In each step, their model computes the optimal dictionary over the similarity matrix after computing the orthogonal mapping. They propose the following key improvements in the dictionary induction step to make the self-learning more robust.

- *Stochastic dictionary induction*: During dictionary induction, they randomly keep some elements in the similarity matrix with some probability, which enables the induced dictionary to vary from the current step to the next. This works as a drop-out method that should help the model to escape poor local optima. Although, in practice, they found that it does not make any difference for most of the language pairs.

- *Frequency based vocabulary cutoff*: To keep the matrix size reasonable, they propose to consider only the most frequent words in the dictionary induction process.

- *CSLS retrieval*: To mitigate the hubness problem (see Section 2.3), they use Cross-domain Similarity Local Scaling (CSLS) (Conneau et al. 2018) for finding the nearest neighbors.

- *Bidirectional dictionary induction*: They propose to induce dictionaries in both directions (source to target and vice versa) and take their concatenation.

(iv) **Final refinement through symmetric re-weighting**: To improve the mapping further, they use a slightly modified version of their earlier multistep framework proposed in Artetxe, Labaka, and Agirre (2018a). In Section 3.2.2, we discuss this step in detail.

In our work, we use a refinement method that combines Procrustes solution and symmetric re-weighting. In contrast to existing methods, our Procrustes solution–based refinement works in the code space, while the symmetric re-weighting method works in the original embedding space. Our method combines the best of both refinement techniques in a way that is mutually advantageous.

Hoshen and Wolf (2018) observe that two sufficiently similar distributions can be aligned correctly with iterative matching methods. In their proposed method, they first align the second moment of the word distributions of two languages, and later refine the alignment iteratively. For aligning the second moment, they project the word vectors to the top P principal components using Principal Component Analysis assuming that some principal axes of variation are similar in many language pairs. Because the word distributions and components of variation are different in languages, projecting to the principal component does not align the languages in general. For this reason, they use a modified version of the Iterative Closest Point (ICP) method, which is popularly used in computer vision for 3D point cloud alignment. They call the method *Mini-Batch Cycle ICP* (MBC-ICP). This method learns the transformation from source to target space and vice versa. They use a cycle constraint to ensure that a word is transformed from one space to another and translated back to the original space without change. In the final step, they use fine-tuning similar to Conneau et al. (2018) by running the Procrustes solution iteratively.

Alvarez-Melis and Jaakkola (2018) cast the unsupervised embedding mapping problem as an optimal transport problem, and exploit the Gromov-Wasserstein distance, which measures how similarities between pairs of words relate across languages.

2.3 Hubness Problem in Similarity Measures

One problem that we have overlooked so far is how to find the nearest neighbor of a source word in the target space. Mikolov, Le, and Sutskever (2013) take the closest target embedding of the mapped source embedding in the target language space using **cosine similarity** as the similarity measure. Dinu, Lazaridou, and Baroni (2015) show that in high dimensional spaces this nearest neighbor finding approach leads to a detrimental phenomenon known as the **hubness** problem. Due to this problem, a few nodes (word embeddings) become hubs, whereas some others become anti-hubs. Hubs are the nodes that are nearest neighbors of many other nodes with high probability. On the other hand, anti-hubs are not nearest neighbors to any node.

To alleviate the hubness problem, Dinu, Lazaridou, and Baroni (2015) propose a method called the **globally corrected neighbor retrieval method**, where instead of returning the nearest neighbor of a (mapped) source embedding, it returns the target embedding for which the source embedding is the nearest neighbor, that is, it reverses the direction of the query. They solve ties by taking the candidate with the highest cosine

similarity with the source embedding. Artetxe, Labaka, and Agirre (2016) termed this approach as **inverted nearest neighbor retrieval**.

Smith et al. (2017) combat the hubness problem by introducing **inverted softmax method**, which is built on the work of Dinu, Lazaridou, and Baroni (2015), and also works by reversing the direction of the query. To find the nearest neighbor, they use the *softmax* function instead of cosine in the similarity computations.

Conneau et al. (2018) consider a bi-partite neighborhood graph, in which each word embedding of a language is connected to its k nearest neighbors in the other language. Let x be the source and y be the target word embeddings, $r_T(x)$ be the average cosine similarity of x to its k nearest neighbors in the target language, and $r_S(y)$ be the average cosine similarity of y to its k nearest neighbors in the source language. The CSLS measure between x and y is computed as:

$$\text{CSLS}(x, y) = 2cos(x, y) - r_T(x) - r_S(y) \tag{4}$$

Among the existing solutions to penalize the similarity scores of hubs, CSLS generally performs better, and has become the standard measure of similarity search. In our approach, we also use CSLS for finding crosslingual nearest neighbors.

3. Our Proposed Approach

Let $\mathcal{X} = \{x_1, \ldots, x_n\}$ and $\mathcal{Y} = \{y_1, \ldots, y_m\}$ be two sets consisting of n and m word embeddings of d-dimensions for a source and a target language, respectively. We assume that $\mathcal{X}$ and $\mathcal{Y}$ are trained independently from monolingual corpora. Our aim is to learn a mapping $f(x)$ in an unsupervised way (i.e., no bilingual dictionary given) such that for every x_i, $f(x)$ corresponds to its translation in $\mathcal{Y}$. Figure 2 shows our overall approach, which has the same sequence of steps as Conneau et al. (2018). More specifically, the steps are:

(i) Induction of seed dictionary through adversarial training.
(ii) Iterative refinement of the initial mapping.
(iii) Application of CSLS for nearest neighbor search.

We propose a novel adversarial autoencoder to learn the initial mapping for inducing a seed dictionary (Section 3.1), and we incorporate existing refinement methods for steps (ii) and (iii) (Section 3.2). Without loss of generality, we use $X \in \mathbb{R}^{n \times d}$ and $Y \in \mathbb{R}^{m \times d}$ to denote the matrices containing the word embeddings of the source and target, respectively. Table 1 summarizes all the notations used throughout the article.

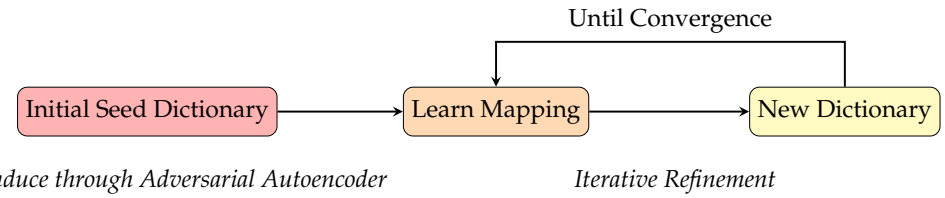


Figure 2
Our framework for unsupervised word translation.

Table 1
Notations used throughout the article.

Notation	Meaning
$x; x_i$	A word embedding in the source language
$y; y_j$	A word embedding in the target language
X	Matrix containing source word embeddings
Y	Matrix containing target word embeddings
$\mathcal{X}; p(x)$	Set (or distribution) of word embeddings in the source language
$\mathcal{Y}; p(y)$	Set (or distribution) of word embeddings in the target language
$E_{\mathcal{X}}$	Encoder for source language autoencoder
$D_{\mathcal{X}}$	Decoder for source language autoencoder
$E_{\mathcal{Y}}$	Encoder for target language autoencoder
$D_{\mathcal{Y}}$	Decoder for target language autoencoder
$\mathcal{Z}_x; q(z_x\|x)$	Distribution of encoded (or code) vectors for source autoencoder
$\mathcal{Z}_y; q(z_y\|y)$	Distribution of encoded (or code) vectors for target autoencoder
Z_x	Matrix containing source code vectors
Z_y	Matrix containing target code vectors
G	Mapper from source codes to target codes
F	Mapper from target codes to source codes
$L_{\mathcal{X}}$	Language discriminator in the source code space
$L_{\mathcal{Y}}$	Language discriminator in the target code space

3.1 Adversarial Autoencoder for Initial Dictionary Induction

Our proposed model as shown in Figure 3 has two autoencoders, one for each language. Each autoencoder comprises an encoder $E_{\mathcal{X}}$ (resp., $E_{\mathcal{Y}}$) and a decoder $D_{\mathcal{X}}$ (resp., $D_{\mathcal{Y}}$). The encoders transform an input x (resp., y) into a latent code z_x (resp., z_y) from which the decoders try to reconstruct the original input. Our autoencoders contain a three-layer encoder and a three-layer decoder with nonlinear transformations in between as shown in Figure 3b. More formally, the encoding-decoding operations of the source autoencoder are defined as:

$$h_1^{E_{\mathcal{X}}} = \texttt{PReLU}(\texttt{Dropout}(\theta_1^{E_{\mathcal{X}}} x_i)) \tag{5}$$

$$h_2^{E_{\mathcal{X}}} = \texttt{PReLU}(\texttt{Dropout}(\theta_2^{E_{\mathcal{X}}} h_1^{E_{\mathcal{X}}})) \tag{6}$$

$$z_{x_i} = \theta_3^{E_{\mathcal{X}}} h_2^{E_{\mathcal{X}}} \tag{7}$$

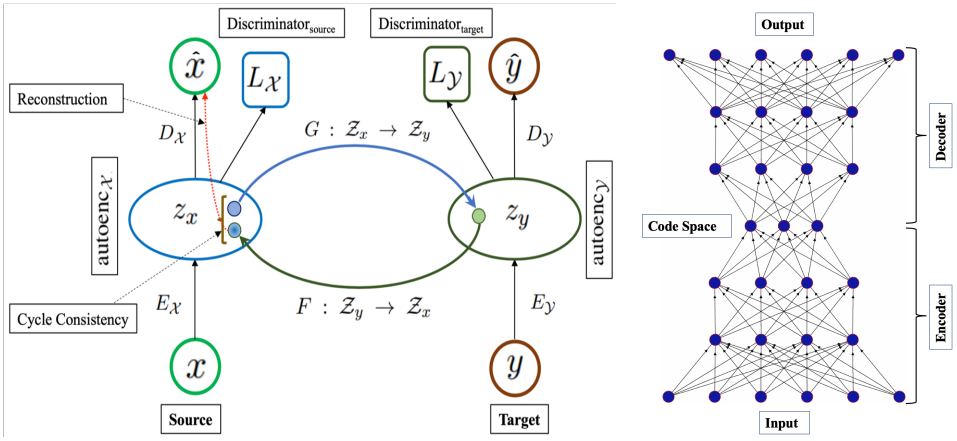
$$h_1^{D_{\mathcal{X}}} = \texttt{PReLU}(\texttt{Dropout}(\theta_3^{D_{\mathcal{X}}} z_{x_i})) \tag{8}$$

$$h_2^{D_{\mathcal{X}}} = \texttt{PReLU}(\texttt{Dropout}(\theta_2^{D_{\mathcal{X}}} h_1^{D_{\mathcal{X}}})) \tag{9}$$

$$\hat{x}_i = \texttt{tanh}(\theta_1^{D_{\mathcal{X}}} h_2^{D_{\mathcal{X}}}) \tag{10}$$

where $\theta_i^{E_{\mathcal{X}}} \in \mathbb{R}^{c_i \times d_i}$ and $\theta_i^{D_{\mathcal{X}}} \in \mathbb{R}^{d_i \times c_i}$ are the parameters of the linear layers in the encoder and the decoder, respectively. We use Parametric Rectified Linear Unit (PReLU) as the nonlinear activation functions. We train the autoencoders with l_2 **reconstruction loss** as defined below.

$$\mathcal{L}_{\text{autoenc}_{\mathcal{X}}}(\Theta_{E_{\mathcal{X}}}, \Theta_{D_{\mathcal{X}}}) = \frac{1}{n} \sum_{i=1}^{n} \|x_i - \hat{x}_i\|^2 \tag{11}$$



(a) Adversarial autoencoder (b) Autoencoder architecture

Figure 3
Our proposed adversarial autoencoder framework for unsupervised word translation.

where $\Theta_{E_{\mathcal{X}}} = \{\theta_1^{E_{\mathcal{X}}}, \theta_2^{E_{\mathcal{X}}}, \theta_3^{E_{\mathcal{X}}}\}$ and $\Theta_{D_{\mathcal{X}}} = \{\theta_1^{D_{\mathcal{X}}}, \theta_2^{D_{\mathcal{X}}}, \theta_3^{D_{\mathcal{X}}}\}$ are the parameters of the encoder and the decoder. The encoder, decoder and the reconstruction loss for the target autoencoder (autoenc$_{\mathcal{Y}}$) are similarly defined.

Let $q(z_x|x)$ and $q(z_y|y)$ be the encoding distributions of the two autoencoders. We use adversarial training to find a mapping between $q(z_x|x)$ and $q(z_y|y)$. This is in contrast with most existing methods (e.g., Conneau et al. (2018), Artetxe, Labaka, and Agirre (2017)) that directly map the distribution of the source word embeddings $p(x)$ to the distribution of the target $p(y)$. As Søgaard, Ruder, and Vulić (2018) pointed out, the isomorphism does not hold in general between the word embedding spaces of two languages. Mapping the latent codes of two languages gives our model more flexibility to induce the required semantic structures in its code space that could potentially yield more accurate mappings.

As shown in Figure 3, we include two linear **mappings** $G : \mathcal{Z}_x \rightarrow \mathcal{Z}_y$ and $F : \mathcal{Z}_y \rightarrow \mathcal{Z}_x$ to project the code vectors (samples from $q(.|.)$) from one language to the other. In addition, we have two language **discriminators**, $L_{\mathcal{X}}$ and $L_{\mathcal{Y}}$. The discriminators are trained to discriminate between the mapped codes and the encoded codes, and the mappers and respective target encoders are jointly trained to fool their respective discriminator. For example, in the case of mapping Z_x to $\mathcal{Z}_y$ by G, the target encoder is $E_{\mathcal{Y}}$. On the other hand, when F maps Z_y to $\mathcal{Z}_x$, $E_{\mathcal{X}}$ is the target encoder. This results in a **three-player** game, where the discriminator tries to identify the origin of a code, and the mapper and the respective target encoder act together to prevent the discriminator from succeeding by making the mapped vector and the encoded vector as similar as possible. In the following, we present the components of our framework.

*Discriminator Loss.*Let $\theta_{L_{\mathcal{X}}}$ and $\theta_{L_{\mathcal{Y}}}$ denote the parameters of the two discriminators, and W_G and W_F are the mapping weight matrices. The loss for the source discriminator $L_{\mathcal{X}}$ is:

$$\mathcal{L}_{L_{\mathcal{X}}}(\theta_{L_{\mathcal{X}}}|W_F, \theta_{E_{\mathcal{X}}}) = -\frac{1}{m}\sum_{j=1}^{m} \log P_{L_{\mathcal{X}}}(\text{src} = 0|F(z_{y_j})) - \frac{1}{n}\sum_{i=1}^{n} \log P_{L_{\mathcal{X}}}(\text{src} = 1|z_{x_i}) \quad (12)$$

where $P_{L_{\mathcal{X}}}(\text{src}|z)$ is the probability according to $L_{\mathcal{X}}$ to distinguish whether z is coming from the source encoder (src $= 1$) or from the target-to-source mapper F (src $= 0$). The discrimination loss $\mathcal{L}_{L_{\mathcal{Y}}}(\theta_{L_{\mathcal{Y}}}|W_G, \theta_{E_{\mathcal{Y}}})$ is similarly defined for the target discriminator $L_{\mathcal{Y}}$ using G and $E_{\mathcal{Y}}$.

Our discriminators have the same architecture as Conneau et al. (2018). Specifically, they are feed-forward networks with two hidden layers of size 2,048 and Leaky-ReLU activations. We apply dropout with a rate of 0.1 on the input to the discriminators. Instead of using 1 and 0, we also apply a smoothing coefficient ($s = 0.2$) in the discriminator loss.

Adversarial Loss. The mappers and encoders are trained jointly to fool their respective discriminators. The adversarial loss for mapper F and encoder $E_{\mathcal{X}}$ can be expressed as:

$$\mathcal{L}_{\text{adv}}(W_F, \theta_{E_{\mathcal{X}}}|\theta_{L_{\mathcal{X}}}) = -\frac{1}{m}\sum_{i=1}^{m}\log P_{L_{\mathcal{X}}}(\text{src} = 1|F(z_{y_i})) - \frac{1}{n}\sum_{i=1}^{n}\log P_{L_{\mathcal{X}}}(\text{src} = 0|z_{x_i}) \tag{13}$$

The adversarial loss for mapper G and encoder $E_{\mathcal{Y}}$ is defined similarly.

Note that in our framework we consider both the mapper and the target encoder as generators. This is in contrast to existing adversarial methods, which do not use any autoencoder on the target side. The mapper and the target encoder team up to fool the discriminator. This forces the discriminator to improve its skill and vice versa for the generators, forcing them to produce indistinguishable codes through better mapping.

Cycle Consistency and Reconstruction. Adversarial training (introduced above) maps a "bag" of source embeddings to a "bag" of target embeddings, and, in theory, the mapper can match the target language distribution (Goodfellow 2017). However, mapping at the bag level is often insufficient to learn the individual word level mappings. In fact, there are an infinite number of possible mappings that can match the same target distribution. Thus to learn better mappings, we need to enforce more constraints to our objective.

The first form of constraints we consider is **cycle consistency** (Zhu et al. 2017) to ensure that a source code z_x translated to the target language code space, and translated back to the original space remains unchanged, that is, $z_x \rightarrow G(z_x) \rightarrow F(G(z_x)) \approx z_x$. Formally, the cycle consistency loss in one direction can be written as:

$$\mathcal{L}_{\text{cyc}}(W_G, W_F) = \frac{1}{n}\sum_{i=1}^{n}\|z_{x_i} - F(G(z_{x_i}))\| \tag{14}$$

The loss in the other direction ($z_y \rightarrow F(z_y) \rightarrow G(F(z_y)) \approx z_y$) is similarly defined.

In addition to cycle consistency, we include another constraint to guide the mapping further. In particular, we ask the decoder of the respective autoencoder to reconstruct the original input from the back-translated code. We compute this **post-cycle reconstruction loss** for the source autoencoder as follows:

$$\mathcal{L}_{\text{rec}}(\theta_{E_{\mathcal{X}}}, \theta_{D_{\mathcal{X}}}, W_G, W_F) = \frac{1}{n}\sum_{i=1}^{n}\|x_i - D_{\mathcal{X}}(F(G(z_{x_i})))\|^2 \tag{15}$$

The reconstruction loss at the target autoencoder is defined similarly.

Apart from improved mapping, both cycle consistency and reconstruction lead to more stable training in our experiments. Specifically, they help our training to converge

and get around the mode collapse issue (Goodfellow 2017). Because the model now has to translate the mapped code back to the source code and reconstruct the original word embedding, the generators cannot get away by mapping all source codes to a single target code.

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

$$\mathcal{L}_{\text{src}\to\text{tar}} = \mathcal{L}_{\text{adv}} + \lambda_1 \mathcal{L}_{\text{cyc}} + \lambda_2 \mathcal{L}_{\text{rec}} \tag{16}$$

where λ_1 and λ_2 control the relative importance of the three loss components. Similarly we define the total loss for mapping in the opposite direction $\mathcal{L}_{\text{tar}\to\text{src}}$.

The complete objective of our adversarial autoencoder model is:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{src}\to\text{tar}} + \mathcal{L}_{\text{tar}\to\text{src}} \tag{17}$$

3.2 Refinement

The encoders ($E_{\mathcal{X}}$ and $E_{\mathcal{Y}}$) and the mappers (G and F) obtained from our adversarial training give a good initial bilingual dictionary. However, as shown later in our experiments (Section 5), the results with these initial mappings are inferior to the supervised methods. One reason is that with adversarial training, we do not consider the embeddings and the covariance of the features (columns of X, Y, or Z) globally. In the refinement step, our goal is to enrich the initial mapping by considering the global properties of the embedding spaces. Previous studies (Conneau et al. 2018; Artetxe, Labaka, and Agirre 2018b) have shown that refinement procedures after the initial mappings boost the results and make them better or on par with the supervised approach. In our work, we experiment with two types of refinement (or fine-tuning) procedures: (*i*) refinement with the Procrustes solution, and (*ii*) refinement with symmetric re-weighting.

3.2.1 Refinement with Procrustes Solution. Similar to Conneau et al. (2018), we first induce a **seed dictionary** using the learned encoders and mappers from our adversarial training. In order to find the nearest target word (y) of a source word (x) in the translated code space ($\mathcal{Z}_y$), we use the CSLS measure in Equation (4). As described in Section 2.3, CSLS works better than simple cosine similarity in mitigating the hubness problem. It penalizes the nodes that are close to many other nodes in the translated space. To construct the seed dictionary, we compute CSLS for K_1 most frequent source (resp., target) words, and select the translation pairs that are mutual nearest neighbors, that is, we select x-y as a translation pair if and only if y is the nearest neighbor of x in $\mathcal{Z}_y$ and x is the nearest neighbor of y in $\mathcal{Z}_x$.

With the seed dictionary, we apply the Procrustes solution in Equation (3) to improve the initial mappings, G and F. In particular, given the approximate alignment of words from the seed dictionary, we optimize the following objectives:

$$W_G = V_G U_G^T, \text{ where } U_G \Sigma_G V_G^T = \text{SVD}(Z_y^T Z_x) \tag{18}$$

$$W_F = V_F U_F^T, \text{ where } U_F \Sigma_F V_F^T = \text{SVD}(Z_x^T Z_y) \tag{19}$$

We perform this fine-tuning process iteratively: Induce a new dictionary using CSLS on the newly learned mapping, then use the dictionary in Procrustes solution

Algorithm 1: Refinement with Procrustes solution

Input : Two sets of word embeddings: $\mathcal{X}$ and $\mathcal{Y}$.
Output: Updated mappings (G, and F)
1. Load the saved best model ($E_{\mathcal{X}}$, $E_{\mathcal{Y}}$, G, and F).
2. Induce a seed dictionary using CSLS.
3. `// Run Procrustes solution iteratively`
do
(i) Find optimal mappings using Procrustes solution using the dictionary (Eq. 3).
(ii) Take most frequent K_1 words from the source and target, and generate a new dictionary using CSLS.
while *not converge*;

to improve the mapping parameters (see Algorithm 1). For convergence, we use the following criterion: if the difference between the average similarity scores of two successive iteration steps is less than a certain threshold (we use 10^{-6}), then stop the refinement process. Note that unlike Conneau et al. (2018), who use original word embeddings; our refinement with Procrustes solution uses the latent codes for both the Procrustes solution and the dictionary induction.

3.2.2 Refinement with Symmetric Re-weighting. In a different line of research, Artetxe, Labaka, and Agirre (2018a, 2018b) perform the refinement and dictionary induction steps by mapping both the source and target embeddings into a common space through orthogonal transformation. In addition to the Procrustes solution, they also use **symmetric** re-weighting to transform the original source and target embeddings to a common space. Because we apply Procrustes solution in the code space ($\mathcal{Z}_{\mathcal{X}}$ and $\mathcal{Z}_{\mathcal{Y}}$), to get the best of both representation types, in our approach we apply symmetric re-weighting to the original embeddings ($\mathcal{X}$ and $\mathcal{Y}$). This refinement works in three steps: (a) embedding whitening, (b) orthogonal mapping, and (c) embedding de-whitening.

(a) Embedding whitening. After performing the length normalization (across rows) and mean-centering (across columns) of the original embedding matrices X and Y, they are whitened by applying a linear transformation with the corresponding whitening matrices, W_x and W_y:

$$X_{\text{whitened}} = XW_x \tag{20}$$

$$Y_{\text{whitened}} = YW_y \tag{21}$$

where $W_x = (X^TX)^{-\frac{1}{2}}$ and $W_y = (Y^TY)^{-\frac{1}{2}}$. Whitening makes the different features of the embeddings have unit variance and zero covariance (i.e., become uncorrelated among themselves).

(b) Orthogonal transformation. In the second step, we perform orthogonal transformation of the whitened matrices with symmetric re-weighting. More specifically, we first compute the singular value decomposition: $USV^T = (X^D_{\text{whitened}})^T Y^D_{\text{whitened}}$, where X^D_{whitened} and Y^D_{whitened} are the whitened embeddings of the induced dictionary entries D from

the previous step. Then we perform orthogonal transformation with symmetric re-weighting as follows:

$$X_{\text{orthogonal}} = X_{\text{whitened}} U S^{\frac{1}{2}} \tag{22}$$

$$Y_{\text{orthogonal}} = Y_{\text{whitened}} V S^{\frac{1}{2}} \tag{23}$$

Note that this step transforms the embeddings into a common space, where they can be compared. However, because they are whitened, they do not represent the original covariance in the feature distributions. Thus, we need a de-whitening step.

(c) Embedding de-whitening. After orthogonal transformation, we de-whiten the transformed matrices to restore the original variance in every direction. Specifically, we perform:

$$X_{\text{w}} = X_{\text{orthogonal}} W_{\text{xdewhitened}} \tag{24}$$

$$Y_{\text{w}} = Y_{\text{orthogonal}} W_{\text{ydewhitened}} \tag{25}$$

where $W_{\text{xdewhitened}} = U^T (X^T X)^{\frac{1}{2}} U$ and $W_{\text{ydewhitened}} = V^T (Y^T Y)^{\frac{1}{2}} V$.

With the transformed de-whitened embeddings, we can now measure the similarity between any $x \in X_w$ and $y \in Y_w$, and thereby induce a synthetic dictionary by finding the nearest neighbor of x in $\mathcal{Y}_w$ using CSLS Equation (4). During dictionary induction, we only consider the K_2 most frequent words from the source and the target languages and retain only the mutual nearest neighbors. This process is then iterated to refine the initial mappings and the induced dictionary. Algorithm 2 shows the whole process in pseudocode.

3.2.3 Combining Procrustes Solution and Symmetric Re-weighting. In our approach, we combine the two refinement methods to get the best of both—Procrustes solution on the code space and symmetric re-weighting on the original embedding space. We sequentially apply the two refinement procedures (Algorithms 1 and 2) by iteratively

Algorithm 2: Refinement with Symmetric Re-weighting

Input : Two sets of word embeddings: $\mathcal{X}$ and $\mathcal{Y}$.
1. Load the saved best model ($E_{\mathcal{X}}$, $E_{\mathcal{Y}}$, G, and F).
2. Induce a seed dictionary using CSLS.
3. `// Run symmetric re-weight iteratively`
do
- (*i*) Preprocess embeddings by length normalization and mean-centering.
- (*ii*) Whiten the embeddings (Eq. 20–21).
- (*iii*) Transform the whitened embeddings through orthogonal mappings (Eq. 22–23).
- (*iv*) Apply de-whitening on the transformed embeddings (Eq. 24–25).
- (*v*) Take most frequent K_2 words from source and target, generate a new dictionary using CSLS

while *not converge*;

performing the same two steps for each method: Induce a synthetic dictionary and refine the mappers.

We first induce a seed dictionary from adversarial training by considering the K most frequent words. To optimize the G and F mappings, we apply the Procrustes solution using the induced dictionary. We then use CSLS to find a new synthetic dictionary, which in turn is used to refine the mappings. We continue this process on the code space until convergence.

After the convergence of the refinement procedure, we take the updated mappings. Using the learned encoders from adversarial training and the newly updated mappings, we induce a new seed dictionary to apply the symmetric re-weighting procedure. We follow the three steps in Section 3.2.2 to transform the original embeddings into a common space. From these transformed source and target embeddings, we induce a new dictionary using CSLS. We apply this refinement procedure iteratively on the original embedding space until convergence.

3.3 Training Procedure and Translation Process

We present the training procedure of our model and the overall word translation process in Algorithm 3. We first pre-train the autoencoders separately on monolingual embeddings (Step 1). This pre-training is required to induce word semantics (and relations) in the latent code space.

We start adversarial training (Step 2) by updating the discriminators for *n_critics* 5 times, each time with a random batch. Then we update the generators (the mapper and target encoder) on the adversarial loss. The mappers then go through two more updates, one for cycle consistency and another for post-cycle reconstruction. The autoencoders (encoder-decoder) in this stage get updated only on the post-cycle reconstruction loss. We also apply the orthogonalization update to the mappers following Conneau et al. (2018) with $\beta = 0.01$.

Our training setting is similar to Conneau et al. (2018), and we apply the similar pre- and post-processing steps. We use stochastic gradient descent with a batch size of 32, a learning rate of 0.1, and a decay of 0.95.

For selecting the best model, we use the **unsupervised validation criterion** proposed by Conneau et al. (2018), which correlates highly with the mapping quality. In this criterion, 10,000 most frequent source words along with their most probable translations in the target space are considered. We use CSLS to find the most probable translation of a source word. The average cosine similarity between these pseudo translations is considered as the validation metric for model selection. For refinement, we follow the methods described in Section 3.2.

4. Experimental Settings

Following the tradition, we evaluate our approach on word translation (*a.k.a.* **bilingual lexicon induction**) task, which measures the accuracy of the predicted dictionary to a gold standard dictionary. In the following, we describe the data sets and the baselines used in our experiments.

4.1 Data Sets

To demonstrate the effectiveness of our method, we evaluate our models on two different data sets. The first one is from Conneau et al. (2018), which consists of FastText

Algorithm 3: Unsupervised word translation with cycle-consistent adversarial autoencoder

Input : Two sets of word embeddings: $\mathcal{X}$ and $\mathcal{Y}$
`// Initial autoencoder training`
1. Train autoenc$_{\mathcal{X}}$ and autoenc$_{\mathcal{Y}}$ separately for some epochs on monolingual embeddings (Eq. 11);
`// Adversarial training`
2. **for** *n_epochs* **do**
  for *n_iterations* **do**
    `// Critic update`
    for *n_critics* **do**
      (*i*) Sample a batch from $\mathcal{X}$ and $\mathcal{Y}$
      (*ii*) Update discriminators ($L_{\mathcal{X}}$, $L_{\mathcal{Y}}$) (Eq. 12)
    end
    (*a*) Sample a batch from $\mathcal{X}$ as source and $\mathcal{Y}$ as target
    (*b*) Update mapper G and encoder $E_{\mathcal{Y}}$ on adversarial loss to fool $L_{\mathcal{Y}}$ (Eq. 13)
    (*c*) Update mappers G and F on cycle consistency loss (Eq. 14)
    (*d*) Update mappers (G, F) and autoenc$_{\mathcal{X}}$ on post-cycle reconstruction loss (Eq. 15)
    `// Orthogonalize the mappers`
    (*e*) Update weight matrices of mapper G and F using:
$$W_G \leftarrow (1+\beta)W_G - \beta(W_G W_G^T)W_G$$
$$W_F \leftarrow (1+\beta)W_F - \beta(W_F W_F^T)W_F$$
    (*f*) Sample a batch from $\mathcal{Y}$ as source and $\mathcal{X}$ as target and update accordingly (symmetric to (*b*) -(*e*) steps).
  end
  Use *validation criterion* to save the best model.
end
`// Fine-tuning`
3. Load the best model.
`// Iterative Procrustes solution`
for *n_iterations* **do**
  (*a*) Build a synthetic dictionary
  (*b*) Apply the Procrustes solution on the dictionary.
end
`// Symmetric re-weighting`
for *n_iterations* **do**
  (*a*) Build a synthetic dictionary
  (*b*) Apply the symmetric re-weighting for the refinement.
end
`// Test`
4. Test the model on gold bilingual dictionary.

monolingual embeddings of (d =) 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 seven (7) different language pairs with fourteen ($7 \times 2 = 14$) different translation tasks from high- and low-resource languages

2 https://github.com/facebookresearch/MUSE.

from different language families. In particular, we evaluate on English (En) from/to Spanish (Es), German (De), Italian (It), Finnish (Fi), Arabic (Ar), Malay (Ms), and Hebrew (He). Malay and Hebrew are generally considered as low-resource languages. We will refer to this data set as the **Conneau data set**.

We also evaluate on the more challenging data set of Dinu, Lazaridou, and Baroni (2015) and its subsequent extension by Artetxe, Labaka, and Agirre (2018a).[3] This data set contains monolingual embeddings for English, Spanish, German, Italian, and Finnish. According to Artetxe, Labaka, and Agirre (2018b), existing adversarial methods often fail to produce meaningful results on this data set. English, Italian, and German embeddings were trained on WacKy crawling corpora using CBOW (Mikolov et al. 2013), while Spanish and Finnish embeddings were trained on WMT News Crawl and Common Crawl respectively. The CBOW vectors are also of 300 dimensions. We refer to this data set as the **Dinu-Artetxe data set**.

4.2 Baselines and Model Settings

We compare our method with the **unsupervised** models of (*i*) Conneau et al. (2018), (*ii*) Artetxe, Labaka, and Agirre (2018b), (*iii*) Alvarez-Melis and Jaakkola (2018), (*iv*) Xu et al. (2018), and (*v*) Hoshen and Wolf (2018). To evaluate how our unsupervised method compares with methods that rely on a bilingual seed dictionary, we follow Conneau et al. (2018), and compute a **supervised** baseline that uses the Procrustes solution directly on the seed dictionary (5,000 pairs) to learn the mapping function, and then uses CSLS to do the nearest neighbor search. We refer to this baseline as **Procrustes-CSLS**.

We also compare with the supervised approaches of Artetxe, Labaka, and Agirre (2017, 2018a), which to our knowledge are the state-of-the-art supervised systems. For some of the baselines, results are reported from their papers, while for the rest we report results by running the publicly available codes on our machine.

For training our model on all language pairs, the weight for cycle consistency (λ_1) in Equation (16) was always set to 5, and the weight for post-cycle reconstruction (λ_2) was set to 1.[4] The hidden layer dimensions of our encoder and decoder are set to 400 for both the first and second layer. We found the dimension of the code vectors to be crucial (especially for Arabic and low-resource languages), which we set through hyperparameter search. During the dictionary induction process of both refinement procedures, we consider 30,000 most frequent words (value of K_1 and K_2) from the source and target languages.

5. Results

We present our results on high- and low-resource languages from Conneau and Dinu-Artetxe data sets in Tables 2–4. In each case, we present results for three different refinement procedures based on the same seed dictionary induced by our adversarial autoencoder: (*i*) refinement of Conneau et al. (2018) referred to as **Conneau refinement**,

3 https://github.com/artetxem/vecmap/.

4 We did not tune the λ values much, rather we used our initial observation. Tuning λ values might yield even better results.

(*ii*) refinement of Artetxe, Labaka, and Agirre (2018b) referred to as **Artetxe refinement**, and (*ii*) our proposed refinement method that combines Procrustes solution and symmetric re-weighting. Through experiments and analysis, our goal is to assess the following.

(i) Does the unsupervised mapping method based on our proposed adversarial autoencoder model improve over the best existing adversarial method of Conneau et al. (2018) in terms of mapping accuracy and convergence (Section 5.1)?

(ii) How does our unsupervised mapping method compare with other unsupervised and supervised approaches (Section 5.2)?

(iii) Which components of our adversarial autoencoder model attribute to improvements (Section 5.3)?

(iv) What is the impact of different refinement methods in our approach (Section 5.4)?

5.1 Comparison with Conneau et al. (2018)

Because our approach follows the same steps as Conneau et al. (2018), we first compare our proposed model with their model on their data set. Table 2 presents the results for Es, De, It, and Fi, and Table 3 presents the results for Ar, Ms, and He. In the tables, we present the numbers that they reported in their paper (Conneau et al. (2018) (paper)) as well as the results that we get by running their code on our machine (Conneau et al. (2018) (code)). For a fair comparison with respect to the quality of the learned mappings (or induced seed dictionary), in this subsection we only consider the results of our approach that use the refinement procedure of Conneau et al. (2018).

In Table 2, we see that our **Adversarial autoencoder + Conneau refinement** outperforms Conneau et al. (2018) in all the eight translation tasks. For Spanish, German, and Italian, the gains are in the range of 0.3% to 1.3%. The improvement is much higher (about 11%) for English to Finnish. Also notice that for Finnish to English, our method gives 64.0% word translation accuracy (P@1), whereas the method of Conneau et al. (2018) fails to converge for this task.

Our method is also superior to theirs for the Arabic and low-resource language pairs (Ms and He) in Table 3. Here our method gives consistent gains ranging from 2.3% to 4.5%. Note specifically that Malay (Ms) is a low-resource language, and FastText contains word vectors for only 155K Malay words. We found their model to be very fragile for En from/to Ms and does not converge at all for Ms→En. We ran their code 10 times for Ms→En but failed every time. Compared with that, our method is more robust and converged most of the time we ran.

If we compare our *Adversarial autoencoder + Conneau refinement* with Conneau et al. (2018) on the Dinu-Artetxe data set in Table 4, we see here also our method performs better than their method in all the eight translation tasks. In this data set, our method shows more robustness compared to their method. For example, their method had difficulties in converging for En from/to Es, De, and Fi translation tasks. For example, their model converges only 2 times out of 10 attempts for En→Es, while for Es→En, En↔De (both directions) and En↔Fi (both directions), it did not converge a single

Table 2
Word translation accuracy (P@1) of **Es, De, It, and Fi** on the **Conneau data set** using **FastText** embeddings (trained on **Wikipedia**). '-' indicates the authors did not report the number. ** indicates failure to converge.

	En-Es		En-De		En-It		En-Fi	
	→	←	→	←	→	←	→	←
Supervised Baselines								
Artetxe, Labaka, and Agirre (2017)	81.2	83.5	72.9	72.5	76.1	77.5	40.8	57.1
Artetxe, Labaka, and Agirre (2018a)	80.5	83.8	73.6	73.5	77.1	79.3	48.9	64.6
Procrustes-CSLS	82.4	83.9	75.3	72.7	78.1	78.1	46.7	58.6
Unsupervised Baselines								
Alvarez-Melis and Jaakkola (2018)	81.7	80.4	71.9	72.8	78.9	75.2	–	–
Xu et al. (2018)	79.5	77.8	69.3	67.0	73.5	72.6	–	–
Artetxe, Labaka, and Agirre (2018b)	82.2	84.4	74.9	74.1	78.9	79.5	**49.8**	63.5
Hoshen and Wolf (2018)	82.1	84.1	74.7	73.0	77.9	77.5	43.6	56.4
Conneau et al. (2018) (paper)	81.7	83.3	74.0	72.2	–	–	–	–
Conneau et al. (2018) (code)	82.3	83.7	74.2	72.6	78.3	78.1	38.4	**
Our Unsupervised Approach								
Adversarial autoencoder +								
Conneau refinement	82.6	84.5	75.5	73.9	78.8	78.9	49.1	64.0
Artetxe refinement	82.7	84.7	75.4	74.1	79.2	79.4	49.4	64.6
Our combined refinement	**83.0**	**85.2**	**76.2**	**74.7**	**79.3**	**80.3**	**49.8**	**65.7**

Table 3
Word translation accuracy (P@1) of **Arabic** and **low-resource (Ms, He)** languages on the **Conneau data set** using **FastText** embeddings. ** indicates failure to converge.

	En-Ar		En-Ms		En-He	
	←	→	←	→	←	←
Supervised Baselines						
Artetxe, Labaka, and Agirre (2017)	24.8	45.3	38.8	41.6	32.7	52.1
Artetxe, Labaka, and Agirre (2018a)	**41.2**	55.2	**55.1**	51.7	**47.6**	58.0
Procrustes-CSLS	34.5	49.7	47.3	46.6	39.2	54.1
Unsupervised Baselines						
Hoshen and Wolf (2018)	34.4	49.3	**	**	36.5	52.3
Artetxe, Labaka, and Agirre (2018b)	33.2	52.8	49.0	49.7	43.8	57.5
Conneau et al. (2018) (code)	29.3	47.6	46.2	**	36.8	53.1
Our Unsupervised Approach						
Adversarial autoencoder +						
Conneau refinement	33.8	49.9	49.5	48.6	41.1	56.8
Artetxe refinement	38.3	54.1	54.0	54.4	44.9	58.1
Our combined refinement	38.6	**55.7**	**54.8**	**55.2**	46.1	**58.6**

time in 10 attempts. Compared with that, our method was more robust and converged most of the time. In Section 5.3, we compare our model with Conneau et al. (2018) more rigorously by evaluating them with and without fine-tuning and measuring their performance on P@1, P@5, and P@10.

These comparisons with Conneau et al. (2018) using the same refinement method demonstrate that our approach of performing the crosslingual mapping in the code

space is more effective and can learn more robust mapping functions. Figures 4a and 4b, respectively, show the t-SNE plots for English to Finnish translation on the **Conneau data set** for the model of Conneau et al. (2018) and our model. Notice that the English words and their Finnish translations are better mapped in our code space compared with the mappings in the original embedding space by Conneau et al. (2018).

5.2 Comparison with Other Methods

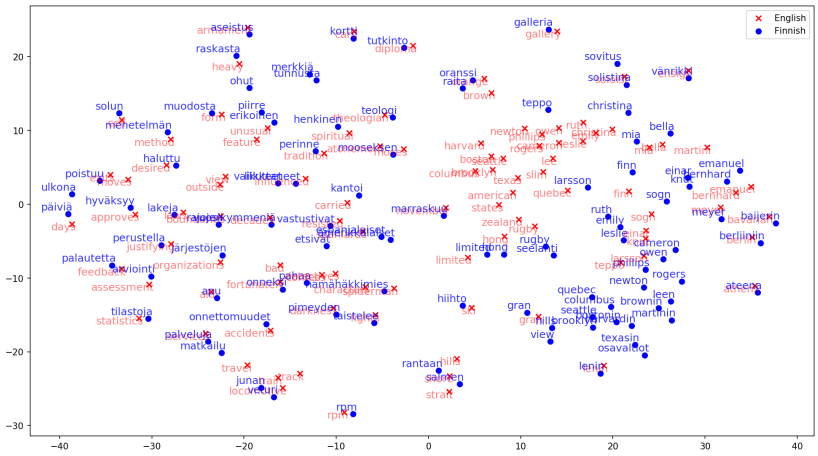
In this section, we compare our model with other state-of-the-art methods that do not follow the same procedure as us and Conneau et al. (2018). For example, Artetxe, Labaka, and Agirre (2018b) do the initial mapping in the similarity space, then they apply a different refinement procedure on that.

Let us first consider the results on the Conneau data set in Table 2. Our method performs better than other methods in all eight translation tasks on this data set. Among the other unsupervised baselines, Artetxe, Labaka, and Agirre (2018b) exhibit better results than others. On the Dinu-Artetxe data set in Table 4, our model achieves better performance than most of the other methods. Only for En from/to Fi does the supervised model of Artetxe, Labaka, and Agirre (2018a) perform better than our method. As we mentioned earlier, this data set is more challenging, where other unsupervised methods except Artetxe, Labaka, and Agirre (2018b) fail to converge in most of the word translation tasks.

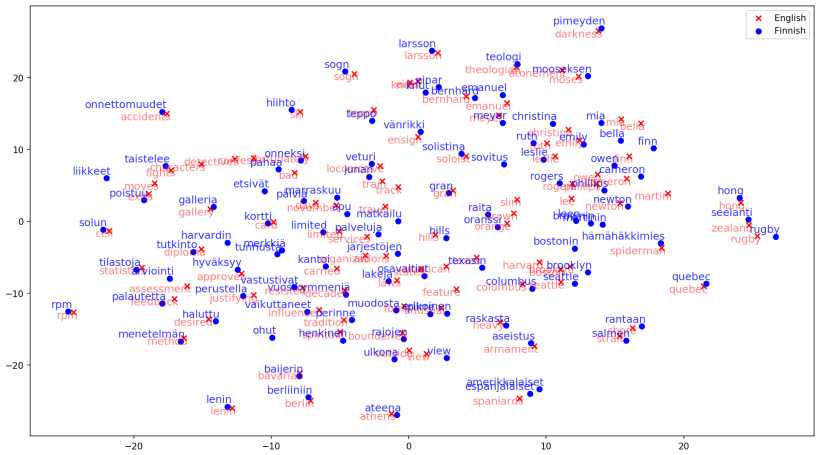
For Arabic and low-resource languages in Table 3, our model exhibits better performance than others in three out of six translation tasks. Only the supervised model of Artetxe, Labaka, and Agirre (2018a) performs better than our method in the rest of the three translation tasks. Here our model gives consistent gains compared with other unsupervised models. For En $\leftrightarrow$ Ms, we see the similar phenomenon that other unsupervised methods except Artetxe, Labaka, and Agirre (2018b) fail to converge.

Table 4
Word translation accuracy (P@1) on the **En-It, En-Es, En-De**, and **En-Fi** data sets of Dinu, Lazaridou, and Baroni (2015), Artetxe, Labaka, and Agirre (2017); All methods use **CBOW** embeddings. ** indicates failure to converge; '-' indicates the authors did not report.

	En-It		En-Es		En-De		En-Fi	
	$\rightarrow$	$\leftarrow$	$\rightarrow$	$\leftarrow$	$\rightarrow$	$\leftarrow$	$\rightarrow$	$\leftarrow$
Supervised Baselines								
Artetxe, Labaka, and Agirre (2017)	43.8	37.2	32.4	27.2	47.4	40.7	30.8	26.2
Artetxe, Labaka, and Agirre (2018a)	45.3	38.5	37.2	29.6	47.2	**44.7**	**33.2**	**36.3**
Procrustes-CSLS	44.9	38.5	33.8	29.3	46.5	42.6	31.8	29.1
Unsupervised Baselines								
Artetxe, Labaka, and Agirre (2018b)	**47.9**	42.3	36.9	31.6	48.3	44.1	32.9	33.5
Hoshen and Wolf (2018)	**	**	**	**	**	**	**	**
Conneau et al. (2018) (paper)	45.1	38.3	–	–	–	–	–	–
Conneau et al. (2018) (code)	44.9	38.7	34.7	**	**	**	**	**
Our Unsupervised Approach								
Adversarial autoencoder +								
Conneau refinement	45.3	39.4	35.2	29.9	46.8	42.6	30.4	31.9
Artetxe refinement	**47.9**	**42.6**	37.5	32.1	47.9	44.1	32.9	33.0
Our combined refinement	47.7	42.3	**38.1**	**32.3**	**48.7**	44.1	32.6	33.2



(a) t-SNE plot for Conneau et al. (2018) model



(b) t-SNE plot for our model

Figure 4
t-SNE plots for En→Fi translation on the **Conneau data set**.

We notice that the unsupervised method of Artetxe, Labaka, and Agirre (2018b) gives better results than other baselines. To understand whether the improvements of their method are due to a better initial mapping or better post-processing, we conducted two additional experiments. In our first experiment, we use their method to induce the initial seed dictionary and then apply iterative Procrustes solution for refinement. Table 5 shows the results. Surprisingly, on both data sets their initial mappings fail to produce any reasonable results. So we suspect that the main gain in Artetxe, Labaka, and Agirre (2018b) comes from their fine-tuning method, which they call *robust self learning*. In the second experiment, we use the initial dictionary induced by our adversarial training and then apply their refinement procedure. Here, for most of the translation tasks, we achieve better results; see the model **Adversarial autoencoder + Artetxe refinement** in Tables 2–4. This shows that the initial dictionary generated by our model is better than their model.

Table 5
Conneau refinement (Iterative Procrustes solution and CSLS) applied to the initial mappings of Artetxe, Labaka, and Agirre (2018b). ** indicates failure to converge.

	En-It		En-Es	
	→	←	→	←
Dinu-Artetxe data set	**	**	**	**
Conneau data set	01.2	01.6	04.7	05.1

5.3 Adversarial Model Dissection

We further analyze our adversarial autoencoder model by dissecting it and measuring the contribution of each novel component that is proposed in this work. We achieve this by *incrementally* removing a new component from our model and evaluating it on different translation tasks. In order to better understand the contribution of each component, we evaluate each model by measuring its **P@1**, **P@5**, and **P@10 with fine-tuning** and **without fine-tuning**. For fine-tuning, we use the **Conneau refinement**. In case of **without fine-tuning**, the models apply the CSLS directly on the mappings learned from the adversarial training, that is, no Procrustes solution–based refinement is done after the adversarial training. This set-up allows us to compare our model directly with the model of Conneau et al. (2018), putting the effect of fine-tuning aside.

Table 6 presents the ablation results for En-Es, En-De, and En-It in both directions. The first row (**Conneau-18**) presents the results of Conneau et al. (2018) that uses adversarial training to map the *word embeddings*. The next row shows the results of **our full** model. The subsequent rows incrementally detach one component from our model. For example, **- Enc. adv** denotes the variant of our model where the target encoder is not trained on the adversarial loss ($\theta_{E_{\mathcal{X}}}$ in Eq. 13); **- - Recon** excludes the post-cycle reconstruction loss from **- Enc. adv**, and **- - - Cycle** excludes the cycle consistency from **- - Recon**. Thus, **- - - Cycle** is a variant of our model that uses only adversarial loss to learn the mapping. However, it is important to note that in contrast to Conneau et al. (2018), our mapping is performed at the code space.

As we compare our full model with the model of Conneau et al. (2018) in the *without fine-tuning* setting, we notice large improvements in all measures across all data sets: 5.1–7.3% in En→Es, 3–6% in Es→En, 3.4–4.3% in En→De, 1–3% in De→En, 3.4–4.3% in En→It, and 0.3–3.7% in It→En. These improvements demonstrate that our model finds a better mapping compared to Conneau et al. (2018). Among the three components, the cycle consistency is the most influential one across all languages. Training the target encoder adversarially also gives a significant boost. The reconstruction has less impact. If we compare the results of - - - *Cycle* with *Conneau-18*, we see sizeable gains for En-Es in both directions. This shows the benefits of mapping at the code level.

Now let us turn our attention to the results with fine-tuning. For a fair comparison with respect to the learned mappings, here we only consider the results of our approach that uses the fine-tuning (refinement) procedure of Conneau et al. (2018). Here also we see gains across all data sets for our model, although the gains are not as verbose as before (about 1% on average). However, this is not surprising as it has been shown that *iterative fine-tuning* with Procrustes solution is a robust method that can recover many errors made in the initial mapping (Conneau et al. 2018). Given a good enough initial

Table 6
Ablation study of our adversarial autoencoder model on the **Conneau data set**.

	En→Es			Es→En			En→De			De→En			En→It			It→En		
	P@1	**P@5**	**P@10**	**P@1**	**P@5**	**P@10**	**P@1**	**P@5**	**P@10**	**P@1**	**P@5**	**P@10**	**P@1**	**P@5**	**P@10**	**P@1**	**P@5**	**P@10**
	Without Fine-Tuning																	
Conneau-18	65.3	73.8	80.6	66.7	78.3	80.8	61.5	70.1	78.2	60.3	70.2	77.0	64.8	75.3	79.4	63.8	77.1	81.8
Our (full)	71.8	81.1	85.7	72.7	81.5	83.8	64.9	74.4	81.8	63.1	71.3	79.8	68.2	78.9	83.7	67.5	77.6	82.1
- Enc. adv	70.5	79.7	83.5	71.3	80.4	83.3	63.7	73.5	79.3	62.6	70.5	79.0	67.6	77.3	82.7	66.2	78.3	82.5
- - Recon	70.1	78.9	83.4	70.8	81.1	83.4	63.1	73.8	80.5	62.2	71.7	78.7	66.9	79.7	82.1	64.8	78.6	82.1
- - - Cycle	66.8	76.5	82.1	67.2	79.9	82.7	61.4	69.7	77.8	60.1	69.8	76.5	65.3	75.1	78.9	64.4	77.6	81.7
	With Fine-Tuning																	
Conneau-18	82.3	90.8	93.2	83.7	91.9	93.5	74.2	89.0	91.5	72.6	85.7	88.8	78.3	88.4	91.1	78.1	88.2	90.6
Our (full)	82.6	91.8	93.5	84.5	92.3	94.3	75.5	90.1	92.9	73.9	86.5	89.3	78.8	89.2	91.9	78.9	88.9	91.1
- Enc. adv	82.5	91.6	93.5	84.3	92.1	94.3	75.4	89.7	92.7	73.5	86.3	89.2	78.4	89.0	91.8	78.1	88.7	91.0
- - Recon	82.5	91.6	93.4	84.1	92.2	94.3	75.3	89.4	92.6	73.2	85.9	89.0	78.2	89.1	91.9	78.2	88.8	91.2
- - - Cycle	82.4	91.0	93.1	83.6	92.2	94.0	74.3	89.7	92.6	72.7	86.1	89.1	77.8	89.2	91.8	77.4	88.3	90.8

mapping, the measures converge nearly to the same point even though the differences were comparatively more substantial initially; for example, notice the scores are very similar for P@5 and P@10 measures after fine-tuning.

5.4 Analysis of the Refinement Procedures

Given the initial mappings and the resulting seed dictionary from our adversarial autoencoder model, we further analyze the impact of different refinement methods. In Sections 3.2.1 and 3.2.2, we described two refinement techniques to fine-tune the initial mappings, namely, refinement with Procrustes solution and with symmetric re-weighting. We then proposed a refinement method (Section 3.2.3) that combines these two techniques. In this section, we analyze the effect of each of these techniques and compare them with the refinement process of Artetxe, Labaka, and Agirre (2018b). Tables 7, 8, and 9 present the results of our experiments on the two data sets using the same initial dictionary for each language pair.

As discussed before, the main phenomenon behind the impressive results of Artetxe, Labaka, and Agirre (2018b) is their refinement procedure. After the robust self-learning step, they perform symmetric re-weighting. We investigate the efficacy of this final symmetric re-weighting step. For the languages on the Conneau data set in Table 7, we see that symmetric re-weighting (**Artetxe refinement**) boosts the results of robust self-learning by 0.1–0.7% for Spanish, German, and Italian. The improvement is more vivid for En from/to Fi, gaining 2.7% and 3.8%, respectively. For Arabic and low-resource languages in Table 8, we see the similar phenomenon—symmetric re-weighting improves the result ranging from 1.2% to 3.9%. On the Dinu-Artetxe data set in Table 9, we see that symmetric re-weighting step (**Artetxe refinement**) on top of robust self-learning improves the results by 1.0–3.9%.

Now if we look at the results of **our refinement** methods in Tables 7–9, we see that our combined method outperforms the individual ones in each of the fourteen (14) translation tasks. If only the Procrustes solution is used for refinement on the Conneau data set, the results lag behind the combined method in the range of 0.4–1.7% for the languages in Table 7 and of 1.8–6.6% for Arabic and low-resource language pairs in Table 9. On the Dinu-Artetxe data set, we see similar phenomenon—results for

Table 7
Analysis of **refinement methods** applied to the same initial mappings of our adversarial autoencoder on the **Conneau data set** for **Es, De, It**, and **Fi** languages.

	En-Es		En-De		En-It		En-Fi	
	→	←	→	←	→	←	→	←
Artetxe refinement								
Robust self-learning	82.3	84.0	75.3	73.6	78.7	78.9	46.7	60.8
Robust self-learning + Symmetric re-weighting	82.7	84.7	75.4	74.1	79.2	79.4	49.4	64.6
Our refinement								
Procrustes solution	82.6	84.5	75.5	73.9	78.8	78.9	49.1	64.0
Symmetric re-weighting	82.8	84.8	75.7	74.5	79.1	80.0	47.6	64.0
Procrustes solution + Symmetric re-weighting	**83.0**	**85.2**	**76.2**	**74.7**	**79.3**	**80.3**	**49.8**	**65.7**

Table 8
Analysis of **refinement methods** applied to the same initial mappings of our adversarial autoencoder on the **Conneau data set** for **Ar, Ms**, and **He** languages.

	En-Ar		En-Ms		En-He	
	→	←	→	←	→	←
Artetxe refinement						
Robust self-learning	35.6	51.7	52.2	50.5	43.7	56.3
Robust self-learning + Symmetric re-weighting	38.3	54.1	54.0	54.4	44.9	58.1
Our refinement						
Procrustes solution	33.8	49.9	49.5	48.6	41.1	56.8
Symmetric re-weighting	36.1	54.0	54.6	55.0	45.8	57.1
Procrustes solution + Symmetric re-weighting	**38.6**	**55.7**	**54.8**	**55.2**	**46.1**	**58.6**

Table 9
Analysis of **refinement methods** applied to the same initial mappings of our adversarial autoencoder on the **Dinu-Artetxe data set**.

	En-It		En-Es		En-De		En-Fi	
	→	←	→	←	→	←	→	←
Artetxe Refinement								
Robust self-learning	44.5	40.5	36.5	30.6	46.8	42.9	31.5	30.4
Robust self-learning + Symmetric re-weighting	**47.9**	**42.6**	37.5	32.1	47.9	44.1	**32.9**	33.0
Our Refinement								
Procrustes solution	45.3	39.4	35.2	29.9	46.8	42.6	30.4	31.9
Symmetric re-weighting	46.5	42.4	37.5	31.9	48.3	44.1	32.4	32.7
Procrustes Solution + Symmetric re-weighting	47.7	42.3	**38.1**	**32.3**	**48.7**	**44.1**	32.6	**33.2**

refinement with only the Procrustes solution are inferior to the combined refinement method in the range of 1.3–2.9%.

Interestingly, when we use only symmetric re-weighting for refinement, the results are close to the results of the combined method. On average, the gain for the combined

method over the symmetric re-weighting methods is about 0.85% on the Conneau data set. We observe similar trends on the Dinu-Artetxe data set.

5.4.1 Impact of Orthogonality Constraint and Regularization on Symmetric Re-weighting. The above observations tell us that the symmetric re-weighting is a powerful method for refinement. In this section, we investigate symmetric re-weighting–based refinement further. Recall that this refinement approach works in three steps: (*a*) embedding whitening, (*b*) orthogonal mapping, and (*c*) embedding de-whitening (see Section 3.2.2). Artetxe, Labaka, and Agirre (2018a) show the correspondence between symmetric re-weighting and the regression-based transformation (see Section 2.1), which are equivalent under certain conditions. The regression-based formulation of the problem gives us further opportunities to explore other possible ways to improve the mapping. For example, **dimensionality reduction** is one that has been shown to be beneficial in the Canonical Correlation Analysis–based approach (Faruqui and Dyer 2014) and in orthogonal transformation (Smith et al. 2017). The idea is to learn mappings after excluding the features that are not indicative (i.e., have low variance).

We conduct a final set of experiments to see if such dimensionality reduction methods yield any further improvement in our framework. For this, we formulate the optimization problem as a regression (OLS) problem as the following.

$$W_{\text{OLS}} = \min_{W} ||Y - XW||_F \tag{26}$$

Then we add regularizers that **promote sparsity** in W (e.g., L_1-regularization, Elastic Net). More specifically, we optimize the following objectives.

$$W_{\text{LASSO}} = \min_{W} ||Y - XW||_F + \gamma_1 ||W||_1 \qquad \texttt{[OLS with } L_1\texttt{]} \tag{27}$$

$$W_{\text{RIDGE}} = \min_{W} ||Y - XW||_F + \gamma_2 ||W||_2^2 \qquad \texttt{[OLS with } L_2\texttt{]} \tag{28}$$

$$W_{\text{E-NET}} = \min_{W} ||Y - XW||_F + \gamma_1 ||W||_1 + \gamma_2 ||W||_2^2 \qquad \texttt{[OLS with } L_1 + L_2\texttt{]} \tag{29}$$

Where γ_1 and γ_2 are the regularization strength parameters. The idea is that if the weights corresponding to certain features in the embeddings become zero (or close to zero), those features are essentially disregarded when mapping (XW) is computed. We use stochastic gradient descent to find the solution. However, Equation (26) does not enforce orthogonality constraint on W, which is shown to be crucial (Artetxe, Labaka, and Agirre 2016; Smith et al. 2017); we will also see this in our experiments. To enforce orthogonality, we update W using the following equation, which ensures that W stays close to an orthogonal matrix during training (Conneau et al. 2018).

$$W_{\text{ORT}} = (1 + \beta)W - \beta(WW^T)W \tag{30}$$

where W is one of $\{W_{\text{OLS}}, W_{\text{LASSO}}, W_{\text{RIDGE}}, W_{\text{E-NET}}\}$, and $\beta = 0.01$ generally performs well.

Tables 10, 11, and 12 show the results of our experiments on the Conneau and Dinu-Artetxe data sets. The first row in each table shows the results for symmetric re-weighting–based refinement and the remaining rows show the results for regression based solutions. We can observe the benefit of using orthogonality constraint on the mapper. For the language pairs on the Conneau data set in Table 10, adding orthogonality constraint improves the results in the range of 2.2–7.1%, and for the language pairs in Table 11 the improvements are much higher, in the range of 5.4–14.4%. For the

Table 10
Analysis of symmetric re-weighting–based refinement applied to the same initial mappings of our adversarial autoencoder of **Es, De, It**, and **Fi** languages on the **Conneau data set**.

	En-Es		**En-De**		**En-It**		**En-Fi**	
	→	←	→	←	→	←	→	←
Symmetric re-weighting	82.8	84.8	75.7	74.5	79.1	80.0	47.6	64.0
OLS	76.2	81.3	72.4	72.3	76.9	76.6	42.9	60.1
OLS + Orthogonality	82.7	84.6	75.9	74.7	79.5	79.9	47.7	63.8
OLS with L_1 regularizer (LASSO)	75.8	81.4	72.4	72.9	76.8	76.8	43.3	59.8
LASSO + Orthogonality	82.3	84.6	75.6	74.7	79.2	79.8	47.4	62.9
OLS with L_2 regularizer (RIDGE)	75.7	81.1	72.2	72.7	77.3	77.6	42.9	60.2
RIDGE + Orthogonality	82.5	84.5	76.2	74.3	79.3	80.1	47.4	63.3
OLS with L_1 & L_2 regularizers (E-NET)	76.0	80.7	72.2	72.9	77.2	77.6	43.5	61.0
E-NET + Orthogonality	82.7	84.9	75.7	74.7	79.2	80.1	47.3	63.6

Table 11
Analysis of symmetric re-weighting–based refinement applied to the same initial mappings of our adversarial autoencoder of **Ar, Ms**, and **He** languages on the **Conneau data set**.

	En-Ar		**En-Ms**		**En-He**	
	→	←	→	←	→	←
Symmetric re-weighting	36.1	54.0	54.6	55.0	45.8	57.1
OLS	27.2	48.0	39.4	44.2	34.5	51.1
OLS + Orthogonality	33.7	53.2	51.6	53.4	42.6	56.4
OLS with L_1 regularizer (LASSO)	26.5	47.9	37.3	43.1	32.4	51.1
LASSO + Orthogonality	34.3	52.8	51.5	52.6	42.6	56.4
OLS with L_2 regularizer (RIDGE)	25.8	47.3	37.3	42.1	32.9	50.8
RIDGE + Orthogonality	33.7	52.8	51.1	52.8	42.3	56.8
OLS with L_1 & L_2 regularizers (E-NET)	27.3	47.5	39.5	41.6	32.5	51.4
E-NET + Orthogonality	33.8	52.2	51.4	53.5	41.9	56.9

language pairs on the Dinu-Artetxe data set in Table 12, we also see the benefit of adding orthogonality constraint.

Now, if we compare the refinement results of the OLS solution with orthogonality constraint with the symmetric re-weighting (first row) on the same induced seed dictionary, we see that for the language pairs on the Conneau data set in Table 10, a small improvement is visible. On the contrary, for other language pairs on the Conneau data set (Table 11) and the language pairs on the Dinu-Artetxe data set (Table 12), OLS with orthogonality constraint lags behind by about 2% on average.

However, we do not see any significant contribution from the regularizers in Tables 10, 11, and 12. After investigation we found that the values in W are generally quite small (in the range of -0.3 to 0.3). As a result, regularizers that penalize large weight values do not seem to have a significant contribution.

Table 12
Analysis of symmetric re-weighting–based refinement applied to the same initial mappings of our adversarial autoencoder on the **Dinu-Artetxe data set**.

	En-Es		**En-It**		**En-De**		**En-Fi**	
	→	←	→	←	→	←	→	←
Symmetric re-weighting	46.5	42.4	37.5	31.9	48.3	44.1	32.4	32.7
OLS	41.8	36.7	29.0	29.6	41.5	39.3	27.7	26.9
OLS + Orthogonality	45.1	39.5	35.7	31.7	47.6	43.8	30.9	32.3
OLS with L_1 regularizer (LASSO)	42.3	36.9	28.4	30.4	42.3	39.7	26.8	27.1
LASSO + Orthogonality	46.0	40.1	34.8	32.2	47.1	44.2	32.4	32.6
OLS with L_2 regularizer (RIDGE)	41.7	36.5	29.4	29.5	41.9	39.0	28.2	29.4
RIDGE + Orthogonality	46.2	39.9	35.1	32.0	47.8	44.2	32.7	32.7
OLS with L_1 & L_2 regularizers (E-NET)	42.1	36.8	29.3	30.1	42.2	38.2	28.5	28.5
E-NET + Orthogonality	45.6	40.3	35.3	31.6	47.7	44.0	32.5	32.9

6. Conclusions and Future Directions

In this article, we have proposed an adversarial autoencoder framework to learn the crosslingual mapping of monolingual word embeddings of two languages in a completely unsupervised way. In contrast to the existing methods that directly map word embeddings, our method first learns to transform the embeddings into latent code vectors by pretraining an autoencoder.

We apply adversarial training to map the distributions of the source and target code vectors. In our adversarial training, both the mapper and the target encoder are treated as generators that act jointly to fool the language discriminator. To guide the mapping further, we include constraints for cycle consistency and post-cycle reconstruction.

To improve the initial mapping further, we use two iterative refinement methods—Procrustes solution and symmetric re-weighting—sequentially on the seed dictionary induced from the adversarial training. While the Procrustes solution–based refinement operates in the latent code space, symmetric re-weighting works in the original word embedding space.

Through extensive experimentations on six different language pairs comprising high- and low-resource languages from two different data sources, we demonstrate that our adversarial method outperforms the method of Conneau et al. (2018) for all translation tasks in all measures (P@{1,5,10}) across all settings (with and without fine-tuning). Comparison with other existing methods also shows that our method learns better mapping. With an extensive ablation study, we further demonstrated that the cycle consistency is the most important component, followed by the adversarial training of target encoder and the post-cycle reconstruction.

From the in-depth analysis of the refinement procedures, we observe the strength of symmetric re-weighting and the significant effect of orthogonality constraint on it. Our refinement approach that combines both Procrustes solution and symmetric re-weighting achieves the best results across almost all of the translation tasks in two data sets.

The work presented in this article leads to several interesting future directions. In the near future, we plan to incorporate knowledge from the similarity space in our adversarial framework. We also want to use word frequency information in our adversarial model. We would also like to explore the fine-tuning process more rigorously,

especially for low-resource languages where the initial mapping obtained from the adversarial training is not that good. Another interesting future direction is to extend our framework to solve downstream crosslingual applications such as machine translation, named entity recognition, part-of-speech tagging, and parsing.

To the best of our knowledge, our work in this article is the first to give a comprehensive overview of existing methods for unsupervised word translation, which is one of the most emerging topics in crosslingual representation learning. We therefore hope that the codebase[5] released with this article will serve as a benchmark that will facilitate other researchers in pushing the state-of-the-art and in applying bilingual word embeddings to their downstream NLP tasks.

Bibliographic Note

Portions of this work have been published in the NAACL-HLT 2019 conference proceeding (Mohiuddin and Joty 2019). However, this article substantially extends the published work in several ways, most notably: (*i*) we extend our approach by incorporating two types of refinement (fine-tuning) procedures (Section 3.2); (*ii*) alongside evaluating the performance with Conneau et al. (2018) refinement and Artetxe, Labaka, and Agirre (2018b) refinement, we also present the performance of our adversarial model with our proposed refinement procedure (Section 5.2); and (*iii*) we analyze the components of different refinement methods and assess the impact of regularization and orthogonality constraint on refinement methods (Section 5.4). Besides these extensions, most of the article is rewritten to adapt to a journal-style publication.

Acknowledgments

Many thanks to the anonymous reviewers for their insightful comments on the NAACL-HLT 2019 submitted version.

References

Alvarez-Melis, David and Tommi Jaakkola. 2018. Gromov-Wasserstein alignment of word embedding spaces. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1881–1890, Brussels.

Artetxe, Mikel, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294, Austin, TX.

Artetxe, Mikel, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, Vancouver.

Artetxe, Mikel, Gorka Labaka, and Eneko Agirre. 2018a. Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5012–5019, New Orleans.

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

Barone, Antonio Valerio Miceli. 2016. Towards cross-lingual distributed representations without parallel text trained with adversarial autoencoders. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 121–126. Berlin.

Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Conneau, Alexis, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer,

5 https://ntunlpsg.github.io/project/unsup-word-translation/.

and Hervé Jégou. 2018. Word translation without parallel data. In *International Conference on Learning Representations (ICLR)*, Vancouver.

Dinu, Georgiana, Angeliki Lazaridou, and Marco Baroni. 2015. Improving zero-shot learning by mitigating the hubness problem. In *ICLR, Workshop track*, San Diego, CA.

Faruqui, Manaal and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471, Gothenburg.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, Curran Associates, Inc., pages 2672–2680.

Goodfellow, Ian J. 2017. NIPS 2016 tutorial: Generative adversarial networks. *CoRR*, abs/1701.00160.

Hoshen, Yedid and Lior Wolf. 2018. Non-adversarial unsupervised word translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 469–478, Brussels.

Lample, Guillaume, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018a. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations (ICLR)*, Vancouver.

Lample, Guillaume, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018b. Phrase-based & neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Brussels.

Luong, Thang, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 151–159, Denver.

Makhzani, Alireza, Jonathon Shlens, Navdeep Jaitly, and Ian J. Goodfellow. 2015. Adversarial autoencoders. *CoRR*, abs/1511.05644.

Mikolov, Tomas, Quoc V. Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119, Curran Associates, Inc. Lake Tahoe.

Mohiuddin, Tasnim and Shafiq Joty. 2019. Revisiting adversarial autoencoder for unsupervised word translation with cycle consistency and improved training. In *HLT-NAACL*, NAACL'19, pages 3857–3867, Minneapolis, MN.

Ruder, Sebastian, Ivan Vulic, and Anders Sogaard. 2017. A survey of cross-lingual word embedding models. *CoRR*, abs/1706.04902.

Smith, Samuel L., David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *International Conference on Learning Representations (ICLR)*, Toulon.

Søgaard, Anders, Sebastian Ruder, and Ivan Vulić. 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)*, pages 778–788, Melbourne.

Xie, Jiateng, Zhilin Yang, Graham Neubig, Noah A. Smith, and Jaime Carbonell. 2018. Neural cross-lingual named entity recognition with minimal resources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 369–379, Brussels.

Xing, Chao, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *HLT-NAACL*, pages 1006–1011, Denver.

Xu, Ruochen, Yiming Yang, Naoki Otani, and Yuexin Wu. 2018. Unsupervised cross-lingual transfer of word embedding spaces. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2465–2474, Brussels, Belgium.

Zhang, Meng, Yang Liu, Huanbo Luan, and Maosong Sun. 2017a. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1959–1970, Vancouver.

Zhang, Meng, Yang Liu, Huanbo Luan, and Maosong Sun. 2017b. Earth mover's distance minimization for unsupervised bilingual lexicon induction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1934–1945, Copenhagen.

Zhu, Jun Yan, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251, Venice.