Supervised and Nonlinear Alignment of Two Embedding Spaces for Dictionary Induction in Low Resourced Languages

Masud Moshtaghi
Amazon / Manhattan Beach, CA USA
mmasud@amazon.com

Abstract

Enabling cross-lingual NLP tasks by leveraging multilingual word embedding has recently attracted much attention. An important motivation is to support lower resourced languages, however, most efforts focus on demonstrating the effectiveness of the techniques using embeddings derived from similar languages to English with large parallel content. In this study, we present a noise tolerant piecewise linear technique to learn a non-linear mapping between two monolingual word embedding vector spaces. We evaluate our approach on inferring bilingual dictionaries. We show that our technique outperforms the state of the art in lower resourced settings with an average improvement of 3.7% for precision @10 across 14 mostly low resourced languages.

1 Introduction

Aligning two embedding spaces is a key component in many multilingual tasks in natural language processing such as unsupervised word and sentence translation (Ruder et al., 2017) and large-scale dictionary creation (Lample et al., 2018). Large scale dictionaries are used in many NLP tasks, for example, web data cleaning. The main feature of the two most successful methods for this task is based on dictionaries (Xu and Koehn, 2017; Koehn et al., 2018).

Initial success obtained in aligning two embedding spaces has created a flurry of research efforts into improving the task (Nakashole, 2018; Doval et al., 2018; Kementchedjhieva et al., 2018; Chen and Cardie, 2018; Joulin et al., 2018; Hoshen and Wolf, 2018; Artetxe et al., 2018), to name a few published in 2018. There are two main approaches for mapping a monolingual word embedding space into another (Ruder et al., 2017): 1) Supervised methods rely on a few anchor points to learn the matching of two spaces (Lample et al., 2018; Mikolov et al., 2013; Xing et al., 2015; Joulin et al., 2018) and; 2) Unsupervised methods mostly formulate the problem as an adversarial optimization problem (Lample et al., 2018; Zhang et al., 2017; Ruder et al., 2017; Chen and Cardie, 2018; Hoshen and Wolf, 2018). While for high density languages such as Spanish, German, French and Italian the accuracy of unsupervised techniques is comparable, or even higher, than supervised techniques, the accuracy of unsupervised techniques is significantly lower when lower resourced languages are considered. A number of the experiments featured in the recent publications reflect this fact (Søgaard et al., 2018; Irvine and Callison-Burch, 2017).

To place this work in the context of recent advances in the field, we divide the aspects targeted by researchers to improve the precision with some examples in Fig. 1. While some of the improvements in optimization and space-adjustments can be applied to almost (any) model, our goal is to improve the supervised mapping in low resourced setting. Since the conception of linear mapping between two spaces, there are only very small minor improvement in the supervised mapping function itself and largest improvements are obtained by updating the word translation retrieval in the shared space and normalization steps (Artetxe et al., 2018). Therefore, we test our method against the most widely cited and tested method by (Lample et al., 2018), which culminates the previous efforts in the area of the supervised mapping (Mikolov et al., 2013; Xing et al., 2015; Lazaridou et al., 2015; Dinu et al., 2015; Smith et al., 2017; Artetxe et al., 2017). The main idea of these methods is to use Procrustes analysis to determine a linear mapping between two spaces using a few anchor points, i.e., a small dictionary. An iterative method is used to expand the dictionary with the most promising candidates and

re-learn the mapping (Artetxe et al., 2017; Lample et al., 2018). However, the main success of these methods was obtained on linguistically similar languages where a large volume of monolingual data from compatible sources is available. A recent study showed that a simple linear mapping is not enough for low resourced languages (Nakashole, 2018). However, the proposed method in this study, NorMA, came short in solving the problem showing significant gains in mapping precision for Russian-English and Chinese-English, compared not to the best model in (Lample et al., 2018), while underperforming in Spanish-English French-English and German-English. Due to the limited set of experiments and inconsistent results across multiple languages we defer comparison with this method until a more mature version is developed. Here, we only acknowledge that our method achieves 1.5% better precision@1 compared to NorMA's results for Chinese.

The contributions of this work are three-fold. First, we introduce a noise tolerant form of *Generalized Procrustes* (GP) which corrects for rotation, geometrical translation [1] and dilation. Second, we adapt an existing technique for piecewise linear regression to instead perform piecewise linear mapping. This allows subspaces to have different mapping functions. Finally, we conduct a large-scale evaluation and show that our model systematically outperforms the state of the art (Lample et al., 2018) in dictionary inference task with the average gain of 3.7% in precision @10 across 14 languages.

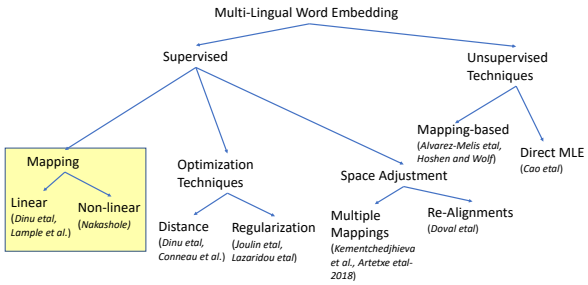

Figure 1: A breakdown of the main area of research focus for dictionary induction task.

[1] Note the difference between the translation of a word and translation as a geometrical transformation of a vector space.

2 Problem Statement

We first provide a formal definition of multilingual word embedding task using a mapping function, then we explain the intuition on why this method can be used to create dictionaries.

2.1 Definition

Given source and target embedding vector spaces V_s and V_o, we are looking for a mapping $V_o \xrightarrow{f} V_s$ that minimize $\|f(V_o) - V_s\|_F$, where $\|.\|_F$ is the Frobenius norm (a matrix norm calculated by the square root of the sum of squares of its elements). Function f performs two tasks, a permutation task to find the matching of vectors and a mapping task to map the permuted vectors in V_o to space spanned by the vectors in V_s. In this general form, the problem is not convex. The supervised solutions for this problem add two assumptions to the problem to make it tractable. First, they assume a set of anchor points are available $\bar{V}_s \subseteq V_s$ and $\bar{V}_o \subseteq V_o$ for which we know a one-to-one matching, i.e., from a small dictionary. The second assumption is that the difference between the two spaces is only in rotation. In this paper, we relax the assumption about f and propose a non-linear mapping, which in addition to the rotational difference, it corrects for dilation and translation between two embedding vector spaces.

2.2 Intuition

We first give an intuition about why we can map a word embedding space into another. Word2Vec (or similar) embedding processes capture the correlation between words (Mikolov et al., 2013) and embed the words in a vector space so that the distances between them are representative of this correlation. If the word embeddings are computed over comparable corpora, a word and its translation have similar correlated words. Hence, these words form similar neighborhood structures in their own monolingual spaces. A reasonable assumption is that the words in this neighborhood should also be a (dictionary) translation of each other. These two assumptions of the similarity of neighborhood structure and the assumption that the neighborhood consists of words that are translations of each other are crucial to the success of mapping methods. The similarity of the corpora used in the initial embeddings in terms of size and content significantly contributes to the validity of the assumptions. This effect is highlighted

in multiple studies (Lample et al., 2018; Søgaard et al., 2018; Grave et al., 2018). That is why the embeddings calculated on high resourced languages, with a similar parallel content to English, are mapped to English embedding space with very high accuracy as the embedding spaces have similar structures. Søgaard *et al.* highlight the language pair, the comparability of the monolingual corpora, and the parameters of the word embedding algorithms as key aspects affecting the performance of unsupervised methods. The performance of supervised methods is also tightly tied to these factors but they are less affected by the virtue of having the extra source of supervision.

If the initial two assumptions hold and since affine mappings, such as orthogonal rotation, do not change the structure of the space, minimizing the distance between the vectors should force the words with the same meaning to end up close to each other. In low resourced languages with a significantly lower amount of data compared to the English corpora structural similarities of the two spaces are not guaranteed leading to weak initial assumptions. This is also true in languages with lower similarity to English such as Slavic and Asian languages (Nakashole, 2018; Nakashole and Flauger, 2018). Nakashole and Flauger showed that word neighborhoods require different linear mappings to correctly capture the word relation. However, the proposed method by (Nakashole, 2018), NorMA, learns a dictionary of neighborhoods and reconstruct the word embedding using a linear combination of the dictionary items which does not guarantee locality in the mapping, , i.e., many dictionary items can contribute to representation of a word.

3 Proposed Method

Our proposed method tackles two problems with the existing mapping techniques. First, instead of only fixing the rotational differences between the two embedding vector spaces, the model also accounts for geometrical translation and dilation. Second, our model, which we call *LLMap*, has different linear mapping functions in subspaces which allows us to have different levels of corrections in different regions of the embedding space. We achieve this goal by incorporating a *Robust Generalized Procrustes* (RGP) mapping into a Locally Linear Model (LLNF).

3.1 Robust Generalized Procrustes Analysis

We borrow the concept Generalized Procrustes and introduce a noise tolerant version, which we name *Robust Generalized Procrustes* (RGP), to solve the first problem with the existing mapping techniques. RGP adds translation and dilation to the vanilla Procrustes. Note that Generalized Procrustes (GP) has been used to mean a different concept in (Kementchedjhieva et al., 2018) referring to a method that maps multiple spaces together.

3.1.1 Procrustes Problem

Procrustes analysis has been used in many different fields for analyzing the relationship between two datasets. The *Orthogonal Procrustes Problem* (*OPP*) (Koschat and Swayne, 1991; Crosilla, 2003), aka Procrustes Rotation, used in current supervised multilingual mapping techniques (Lample et al., 2018; Smith et al., 2017) is defined as,

Definition 1: Given two matrices $\bar{V}_s$ and $\bar{V}_o \in \Re^{n \times p}$ with rank p, the OPP is defined as the following optimization problem.

$$\min_{W} \left\| \bar{V}_o W - \bar{V}_s \right\|_F^2, \text{subject to } W^T W = I_{p \times p} \tag{1}$$

The above optimization function has a closed form solution given by $W^T = UV^*$ where $UDV^* = SVD(\bar{V}_s^T \bar{V}_o)$. The Generalized Procrustes Problem (GPP) formulates the problem as a general linear mapping, correcting for other types of disturbances, i.e., translation and dilation.

Definition 2: Given two matrices $\bar{V}_s$ and $\bar{V}_o \in \Re^{n \times p}$ with rank p, the GPP is defined as the following optimization problem.

$$\begin{aligned} &\min_{W} \left\| \left(\bar{V}_o K W + I_{n \times 1} b \right) - \bar{V}_s \right\|_F^2, \\ &\qquad \text{subject to } W^T W = I_{p \times p} \\ &\textit{where } K \in \Re^{p \times p}(\textit{dilation}),\ b \in \Re^{1 \times p}(\textit{translation}), \\ &\qquad W \in \Re^{p \times p}(\textit{rotation}) \end{aligned} \tag{2}$$

The solutions for the three components of this optimization problem, i.e., K (diagonal (zero off diagonal) positive matrix), b and W are shown in Eq. (3). Let $m_s = mean(\bar{V}_s)$, $m_o = mean(\bar{V}_o)$,

$\bar{V}_o^* = \bar{V}_o - m_o$ and $\bar{V}_s^* = \bar{V}_s - m_s$

$$
\begin{aligned}
W^T &= UV^*, UDV^* = SVD(K^{-1}\bar{V}_s^{*T}\bar{V}_o^*) \\
K_{jj} &= \frac{(\bar{V}_s^*)_j^T(\bar{V}_o^*W)_j}{(\bar{V}_o^*W)_j^T(\bar{V}_o^*W)_j} \\
b &= m_s - m_oKW
\end{aligned}
\tag{3}
$$

where $(.)_j$ denote the j^{th} column vector of the matrix.

The translation factor b in Eq. 3 does not appear in the calculation of rotation and dilation while the other two have interdependencies. Centering all vectors prior to mapping makes the translation vector zero. But it affects the flexibility of subspaces to have different translation factor. Hence, we operate on the original embedding space without centering the vectors.

For the other two factors, Koschar and Swayne (Koschat and Swayne, 1991) suggested using an iterative algorithm starting with $K = I_{p\times p}$ and computing W, then using W updating the values of K and iterate between the two until reaching convergence. We use 3 iterations in all of our experiments as the values converge rapidly. We expect to have noisy embedding vectors, so we use the truncated SVD algorithm to calculate the rotation matrix, i.e., discarding eigenvectors corresponding to small eigenvalues to make the mapping robust to outliers.

In the next section, we introduce how this linear mapping can be incorporated into the LLNF model to learn a piecewise linear mapping between two vector spaces.

3.2 Locally Linear Neural Model for Mapping (LLMap)

The inaccuracies in the embedding process due to lack of data, and inherent language family differences reduce structural similarities in (some) regions of the embedding space. In this case, a global minimization of the distance between the vectors using structure preserving mappings (Procrustes) could create undesired artifacts. We devise a piecewise linear mapping between the two spaces based on a Locally Linear Neural (LLNF) model with soft switching between the neurons (Babuška and Verbruggen, 2003) to increase the flexibility of the models when dealing with different levels of structural similarities in the embedding space. In this work, we update the neuron definition in the standard LLNF model to perform localized mapping using RGP, which guarantees a reduction in terms of the *Sum of Squared Errors* (SSE).

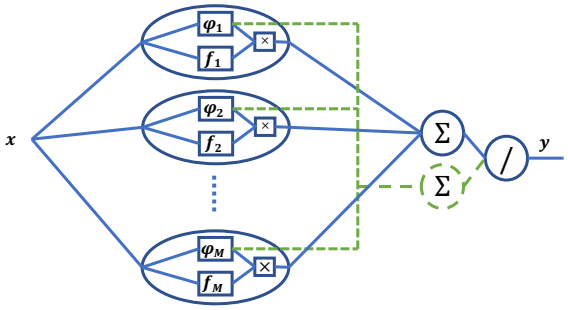

Figure 2: The network structure of an LLNF model. The green dotted line is used to normalize the output of neurons. Each neuron contains a membership function ϕ that controls the contribution of the neuron function f to the final output.

First, we provide a brief introduction to the LLNF model followed by the description of our method, LLMap. LLMap replaces the linear regression function in the LLNF neurons with RGP while exploiting the training algorithm of the LLNF model for parameter estimation.

3.2.1 Introduction to LLNF Models

The LLNF model is a specific form of Radial Basis Function Networks (RBF) with an expanded linear model in each neuron. This model is a general function approximator (Nelles, 2001). An LLNF input-output model with input vector $x \in \Re^p$, output $y \in \Re$ and M neurons has the following form:

$$
y = \frac{\sum_{i=1}^{M} \phi_i(x) f_i(x)}{\sum_{i=1}^{M} \phi_i(x)} \tag{4}
$$

where $f : \Re^p \rightarrow \Re$ is a linear function and $\phi : \Re^p \rightarrow [0\ ,\ 1]$ is called a membership function. Fig. 2 shows a pictorial form of these networks. These networks typically contain one hidden layer. Each neuron has a membership function that controls the output of the neuron generated by a linear regression function inside the neuron. The final output is the weighted average of all the neurons.

The two most common membership functions are the Gaussian and trapezoidal functions. The Gaussian membership function with spherical contours

is defined as

$$\phi(x) = \prod_{j=1}^{p} e^{-\frac{(x_j - c_j)^2}{2\sigma_j^2}}, \tag{5}$$

where $c \in \Re^p$ and $\sigma \in \Re^p$ are the center and standard deviation of the neuron. When p is large the Gaussian function can result in numerical instability and usually a trapezoidal function is preferred in this case. In one dimensional form a trapezoidal function is defined by a quadruple (a, b, c, d) with the following function.

$$\phi(x_j)_j = \begin{cases} 0 & x_j < a \text{ or } x_j \geq d, \\ \frac{x_j - a_j}{b - a_j} & a \leq x_j < b, \\ 1 & b \leq x_j < c, \\ \frac{d - x_j}{d - c} & c \leq x_j < d. \end{cases} \tag{6}$$

An extension to a multidimensional function can be achieved by defining $\phi(x) = \min_j(\phi(x_j)_j)$. We use this membership function our model.

The non-linearity of the LLNF comes from ϕ_i functions. If these functions are predefined, the estimation of f_i parameters is simplified to solving a weighted regression problem. The weight for input vector x at neuron q is $\phi_q(x)/\sum_{i=1}^{M}\phi_i(x)$, i.e., the normalized membership of the vector to the neuron. There are a number of strategies for determining the number of neurons and the ϕ_i functions. A widely used approach is called Locally Linear Model Tree (LoLiMoT), which follows a greedy tree construction algorithm.

LoLiMoT- The LoLiMoT algorithm (Nelles, 2001) creates hyper-rectangular partitions over the input space and centers a neuron in the middle of each partition. The parameters of the neuron are set proportional to the size of the rectangle. In Section 3.2.2, we describe how the parameters of the model are set in our experiments. The algorithm works in a greedy manner. It starts with one hyper-rectangle covering the whole space. Then, it enters a loop to optimize the number of partitions where it finds the rectangle that contributes the most to the estimation error and split the rectangle in the dimension that decreases the estimation error the most as shown in Algorithm 1. The splitting of each neuron is performed using axis-parallel partition of the space into two equal subspaces with a neuron at the center of each subspace. Fig. 3 shows a schematic 2-dimensional view of how the LoLiMoT algorithm partitions the input space.

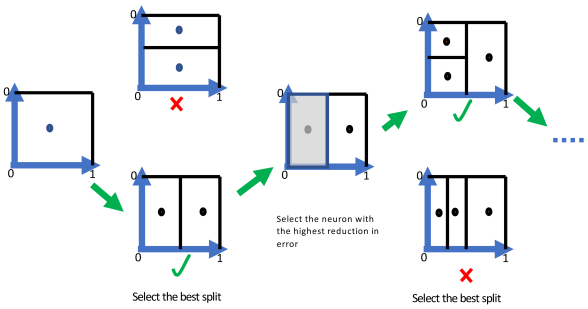

Figure 3: LoLiMoT structure optimization algorithm in 2 dimensions.

3.2.2 LLNF for Mapping

While LLNF models are used for regression and classification (Nelles, 2001), we aim to use these models for mapping by replacing the linear regression component in each neuron with an RGP. We leverage the LoLiMoT algorithm to optimize the network structure. This is achieved by replacing the function f_i in Eq. 4 with a linear mapping function. This results in having a weighted RGP problem to estimate the parameters of f_i. The weighted RGP can also be solved in closed-form (Crosilla, 2003; Koschat and Swayne, 1991) by updating the Eq. 3 to include a diagonal weight matrix L_q where $\ell_{xx} = \phi_q(x_x)/\sum_{i=1}^{M}\phi_i(x)$ is the weight for each input vector at neuron q. The updated formula is written as,

$$\begin{aligned} W_q^T &= UV^*, UDV^* = SVD(K^{-1}(L_q\bar{V}_s^{*T})(L_q\bar{V}_o^*)) \\ K_{jj} &= \frac{(\bar{V}_s^*)_j^T(\bar{V}_o^*W_q)_j}{(\bar{V}_o^*W_q)_j^T(\bar{V}_o^*W_q)} \\ b_q &= m_s - m_oKW_q \end{aligned} \tag{7}$$

The LLNF model in Eq. 4 computes a mapping of each input vector when Eq. 7 is used in place of f_i. With this change, the LoLiMoT algorithm can still be used to optimize the structure of the network.

In this paper, we consider a fixed setup in our experiments for certain hyperparameters of the LLMap to focus the discussion on the main model. These parameters can be potentially tuned to increase the performance. The following setup is used in all of the experiments in this paper.

1. Eq. 7 requires the number of iterations between calculation of rotation and dilation

Input: Matrices $\bar{V}_s$ and $\bar{V}_o$, D = # of dimensions, itr = # of splits
LLMap = `Initialize` (*One neuron covering the entire space*)
while $itr \geq 1$ **do**
  `// Use Eq.(7) for each neuron in LLMap`
  Error = LLMap.`Fit` ($\bar{V}_s$,$\bar{V}_o$)
  foreach $N \in LLMap$ **do**
    `// ` L_q `is a diag. matrix of membership values`
    N.Error = Error*N.L_q
  end
  N_Sel = *N in LLMap with Maximum Error*
  LLMap_Sel = LLMap
  MaxError = Error.`Sum` ()
  foreach $j \in D$ **do**
    LLMap_New = LLMap - N_Sel
    N1, N2 = N_Sel.`Split` (j)
    LLMap_New.`Add` (*N1, N2*)
    Error = LLMap_New.`Fit` ($\bar{V}_s, \bar{V}_o$)
    if *Error*.`Sum` *()* $<$ *MaxError* **then**
      MaxError = Error.`Sum` ()
      LLMap_Sel = LLMap_New
    end
  end
  $itr - -$
  LLMap = LLMap_Sel
end

Algorithm 1: LLMap Algorithm

matrices. We use 3 iterations in all experiments.

2. To define the quadruple (a, b, c, d) for each neuron, we use two parameters (α, δ) and define $a = \alpha - 4\delta, b = \alpha - \delta, c = \alpha + \delta, d = \alpha + 4\delta$. For the first neuron $\alpha = m_o$ (the mean of input vectors) and $\delta = (max\{\bar{V}_o\} - min\{\bar{V}_o\})/4$. This ensures that the activation function of the first neuron is non-zero for all the values in the space.

3. We set the stopping criterion for the LoLiMoT algorithm to 4 neurons. We impose this restriction to allow the model and the data to fit in GPU memory. In addition, the small number of neurons prevents the vanilla LoLiMoT from overfitting and removing the need for a validation set.

4. To calculate the rotation parameter W, we discard eigenvectors in U, V with corresponding eigenvalue smaller than 0.5 up to a maximum of 50 vectors. This maximum number is used to ensure that as the number of neurons increases LLMap does not remove too many eigenvectors.

4 Experiments

The purpose of the experiments is to compare the performance of our algorithm in low resourced languages for dictionary generation in a supervised setting with the best existing supervised technique. To this end, we compare our model with the supervised MUSE algorithm (Lample et al., 2018).

4.1 Data

We use the dictionaries made publicly available through the MUSE library [2]. This library contains 110 bilingual dictionaries of varied sized, mostly with English as the target languages. We also use the 300-dimensional word embeddings pre-trained on Wikipedia using the skip-gram method with default parameters described in (Bojanowski et al., 2017). [3] This library contains 294 languages.

We select 14 languages from low resourced languages including a wide range of languages including Asian and Slavic and European languages plus Spanish. Our selection mix is focused on low resourced and languages with low structural similarity to English. We use 200,000 most frequent tokens in the embedding file.

4.2 Experimental setup

The goal of dictionary inference using multilingual word embedding is to map the monolingual word embedding spaces onto each other and to infer the translation of a word by looking at the neighborhood of that word. We use the *Cross-Domain Similarity Local Scaling* (CSLS) distance introduced in (Lample et al., 2018) to find the top k. Given a source word, v, and its set of translations, V', any sense retrieval measures the percentage of time that at least one $v' \in V'$ is in the top k-neighbors of v. In all senses retrieval the results

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

Language	P@1			P@5			P@10		
	LLMap	MUSE	RGP	LLMap	MUSE	RGP	LLMap	MUSE	RGP
Czech (CS)	28.29	28.37	28.37	**56.92**	55.65	55.88	**65.99**	64.72	64.94
Norwegian (NO)	**32.9**	31.62	31.63	**58.74**	56.13	56.53	**66.23**	63.57	64.01
Dutch (NL)	**42.3**	41.06	41.18	**67.13**	65.43	65.7	**73.73**	72.03	72.49
Chinese (ZH)	**17.4**	14.19	19.51	**43.19**	35.6	44.05	**52.11**	44.62	52.68
Korean (KO)	17.12	17.02	16.81	**35.8**	34.41	34.23	**44.05**	42.69	42.12
Japanese (JA)	**9.46**	2.45	9.97	**20.3**	6.97	20.72	**25.83**	9.85	26.66
Croatian (HR)	**19.29**	18.71	18.87	**46.4**	43.55	44.25	**56.36**	53.38	54.12
Indonesian (ID)	**31.9**	30.37	30.45	**54.63**	51.96	52.4	**62.06**	59.24	59.67
Farsi (FA)	**17.48**	16.69	17.12	**35.36**	33.14	33.7	**42.7**	40.24	40.75
Bulgarian (BG)	21.64	**23.4**	23.02	56.19	55.22	55.35	**67.12**	65.95	66.11
Spanish (ES)	42.36	42.01	42.08	70.75	69.94	70.12	77.32	76.41	76.73
Tamil (TA)	9.79	9.76	10.07	**24.3**	22.34	23.08	**31.38**	28.78	29.8
Hindi (HI)	**16.83**	16.26	16.79	**36.51**	32.88	34.69	**43.91**	39.92	41.85
Bengali (BN)	15.51	15.3	15.95	**39.67**	35.55	37.06	**48.43**	43.94	45.51
Average Improvement (LLMap-MUSE)	**1.07**			**3.36**			**3.70**		

Table 1: The comparison between the proposed methods LLMap and RGP, and the MUSE supervised method. The values are average precision over 10 random 90-10 splits of the dictionaries, statistically significant results between LLMap and MUSE are shown in bold and between LLMap and RGP are underlined.

are measured by the percentage of time each individual $v' \in V'$ appearing in the top k-neighbors of v.

While MUSE supervised was tested for any sense retrieval using a pre-split train-test, the performance of the dictionary inference task for $k > 1$ is better to be measured by all senses retrieval. Any sense retrieval is considered a significantly easier task, e.g. in Spanish precision@5 of the MUSE algorithm goes up from 71% to 91% when any sense is considered. Also, best experimental design practices suggest using cross validation to account for variation in the data.

Considering these shortcomings, we use a different experimental setup. We create 10 random 90-10 splits of the big dictionaries for training and testing of the systems to capture the variations due to random splits. We also report the precision@5 using the pre-split for compatibility with previously published results on this dataset for both any and all senses of a word.

We use double sided paired t-test and p-value at 0.05 to test for statistical significance in the results. We used 4 refinement steps in the supervised MUSE mapping and used the best result on the test data as the performance of the algorithm. For our algorithm, we report the final test result after adding the 4th neuron. We compare the results of RGP and LLMap to quantify the contribution of each addition that we made to improve the mapping.

4.3 Results

Table 1 shows the average of precision over the 10 random splits @$k = 1, 5$ and 10. The bold values are statistically significant results between LLMap and the MUSE supervised method. The RGP column refers to our model without the piecewise mapping, which we discuss later in this section.

In all cases, except Czech (CS) and Bulgarian (BG) @1 where MUSE has a slight edge over LLMap, our method achieved higher precision on average over 10-fold cross-validation than the MUSE algorithm. In the majority of the cases the improvements are statistically significant. We can see the most significant improvements (over 8%) are observed in Japanese (JA) language and Chinese (ZH). The other languages mostly see between 1%-3% improvement in the precision. The average gain in precision @10 sits at 3.7%.

The gains achieved by our method become larger by increasing the k. It shows that LLMap can create a better proximity neighborhood around the words as it is more flexible to handle multiple

Lang	Any Sense		All Senses	
	MUSE	LLMap	MUSE	LLMap
CS	76.66	**78.86**	61.32	**62.17**
NO	80.00	**81.85**	65.57	**67.35**
NL	88.53	**89.33**	69.28	**70.52**
ZH	55.00	**64.77**	41.88	**51.50**
KO	51.94	**54.23**	42.81	**44.40**
JA	3.97	**17.62**	3.22	**14.51**
HR	62.46	**64.70**	52.19	**53.78**
ID	81.20	**85.19**	64.87	**68.20**
FA	53.73	**56.03**	42.64	**44.29**
BG	65.07	**68.13**	59.45	**59.48**
ES	92.10	**92.23**	71.46	**72.11**
TA	28.53	**29.00**	25.60	**26.01**
HI	51.20	**53.93**	38.22	**44.53**
BN	33.80	**36.42**	42.42	**45.10**
AVG	**3.45**		**3.07**	

Table 2: The comparison between the proposed method LLMap and MUSE on pre-split train and test dictionaries for any and all senses recovery by the algorithms for precision@5. Last row shows the average improvement achieved by LLMap over MUSE

senses of a word. In Spanish the gains are not statistically significant at any k, pointing to the fact that in high resourced languages with large similarities with English the mapping cannot create a significant advantage over simpler mappings.

Table 2 shows the precision@5 for the pre-split dictionaries. In all 14 languages the LLMap outperforms the MUSE algorithm for recovering both all senses and any sense of a word with significant gains in Chinese and Japanese. These results are consistent with the more comprehensive cross-validation settings. Note that the any sense recovery is on average higher than all senses, pointing to the same fact the model is better at creating a better neighborhood around words where at least one sense of a word can be recovered.

The improvements that we observed compared to the state of the art are the culmination of two main changes to the baseline OPP in the mapping calculation, namely using RGP and the piecewise linear mapping. A vanilla RGP method is an LLMap model with one neuron. Table 1 shows how much increasing the non-linearity in the mapping by increasing the number of neurons from 1 to 4 contributes to the overall precision. LLMap has statistically significant (underlined values) higher precision than the vanilla RGP in most cases. However, in the two languages with the largest improvements compared to MUSE, RGP accounts for most of the gains. LLMap contributes to an overall average of 1% increase in the precision on top of RGP.

4.4 Discussions

The computational complexity of the mapping increases linearly with the number of neurons. In terms of training, finding the best dimension to split is the most time-consuming part, which increases the complexity of the algorithm from $O(n^2p)$ in OPP to $O(n^2p^2)$ in LLMap. This step requires creating p (number of dimensions) models and estimate the error of each model. In our implementation, each split took approximately 5 minutes to complete on an Amazon AWS EC2 p3.2xlarge. This task is usually done offline as a pre-processing step. Therefore, the training time complexity would not be a barrier to this approach. While MUSE uses a simpler mapping, the iterations require identifying promising candidates to add to the dictionary, which results in a similar 5 minutes run-time on a GPU.

We limit the number of neurons in our algorithm to 4 for experimental purposes. This allows us to have the models, data and top-k calculation on a single GPU for quick (10-15 minutes) run-time.

The framework presented in this work, is a general framework that can accommodate other mappings and improvements to the distance functions. However, so far to the best of our knowledge, structure preserving mapping such GP outperforms non-linear mappings that change the structure of the underlying space. Regularized and weighted mapping functions that do not have closed-form solutions can also be incorporated in this structure with a back-propagation optimization technique. In this work, we focused our discussion on the generalized mapping function as opposed to other areas of improving this task.

4.5 Comments

We have added this section to explain the The reviewers requested comparison of LLMap with a stronger baseline to compare the supervised mapping, suggesting vanilla VecMap (Artetxe et al., 2018) as a potential baseline. In the introduction, we argued that in terms of mapping function MUSE has the strongest and most widely applicable mapping function. Most of improvements after this work, including the ones in VecMap,

are obtained by updating the translation retrieval and/or normalization around the mapping function. To ensure the validity of our argument, we ran VecMap with normalization, whitening and reweighting turned off and it produced nearly identical numbers to MUSE when CSLS is used as the retrieval method.

5 Conclusion

In this research, we introduced a non-linear mapping between two embedding spaces for dictionary induction. The embedding space created for low resourced languages are noisier than their English counterpart and introduce challenges for simple rotational mapping used in the current techniques. We showed that our piecewise linear model systematically improves the precision of existing Procrustian mapping for dictionary induction using in a large-scale cross-validation setting by over 3% on average. This approach has applications beyond the scope of this paper and can be used in any applications requiring solving OPP or GPP.

References

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

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations. In *AAAI Conference on Artificial Intelligence*.

Robert Babuška and Henk Verbruggen. 2003. Neuro-fuzzy methods for nonlinear system identification. *Annual Reviews in Control*, 27(1):73 – 85.

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

Xilun Chen and Claire Cardie. 2018. Unsupervised multilingual word embeddings. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 261–270, Brussels, Belgium.

Fabio Crosilla. 2003. Procrustes analysis and geodetic sciences. In Schwarze V.S. Grafarend E.W., Krumm F.W., editor, *Geodesy-The Challenge of the 3rd Millennium*, pages 287–292. Springer, Berlin, Heidelberg.

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

Yerai Doval, Jose Camacho-Collados, Luis Espinosa Anke, and Steven Schockaert. 2018. Improving cross-lingual word embeddings by meeting in the middle. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 294–304, Brussels, Belgium.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Language Resources and Evaluation Conference*.

Yedid Hoshen 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, Belgium. Association for Computational Linguistics.

Ann Irvine and Chris Callison-Burch. 2017. A comprehensive analysis of bilingual lexicon induction. *Computational Linguistics*, 43(2):273–310.

Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou, and Edouard Grave. 2018. Loss in translation: Learning bilingual word mapping with a retrieval criterion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2979–2984, Brussels, Belgium. Association for Computational Linguistics.

Yova Kementchedjhieva, Sebastian Ruder, Ryan Cotterell, and Anders Søgaard. 2018. Generalizing Procrustes analysis for better bilingual dictionary induction. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 211–220, Brussels, Belgium. Association for Computational Linguistics.

Philipp Koehn, Huda Khayrallah, Kenneth Heafield, and Mikel Forcada. 2018. Findings of the wmt 2018 shared task on parallel corpus filtering. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 726–739, Belgium, Brussels. Association for Computational Linguistics.

Martin A. Koschat and Deborah F. Swayne. 1991. A weighted procrustes criterion. *Psychometrika*, 56(2):229–239.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *Sixth International Conference on Learning Representations*.

Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. 2015. Hubness and pollution: Delving into

cross-space mapping for zero-shot learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural*, pages 270–280.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119.

Ndapa Nakashole. 2018. NORMA: Neighborhood sensitive maps for multilingual word embeddings. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 512–522, Brussels, Belgium. Association for Computational Linguistics.

Ndapa Nakashole and Raphael Flauger. 2018. Characterizing departures from linearity in word translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, page 221–227, Melbourne, Australia. Association for Computational Linguistics.

Oliver Nelles. 2001. *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models.* Springer, Berlin, Heidelberg.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2017. A survey of cross-lingual word embedding models. *arXiv preprint arXiv:1706.04902v2*.

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

Anders Søgaard, 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, Australia. Association for Computational Linguistics.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of NAACL*, Los Angeles, CA.

Hainan Xu and Philipp Koehn. 2017. Zipporah: a fast and scalable data cleaning system for noisy web-crawled parallel corpora. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2945–2950, Copenhagen, Denmark. Association for Computational Linguistics.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Adversarial training for unsupervised bilingual lexicon induction. In *the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1959–1970, Vancouver, Canada.