Unsupervised Multilingual Alignment using Wasserstein Barycenter

by

Xin Lian

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science (Co-Operative Program)

Waterloo, Ontario, Canada, 2020

Author’s Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

We investigate the language alignment problem when there are multiple languages, and we are interested in finding translation between all pairs of languages. The problem of language alignment has long been an exciting topic for Natural Language Processing researchers. Current methods for learning cross-domain correspondences at the word level rely on distributed representations of words. Therefore, the recent development in the word computational linguistics and neural language modeling has led to the development of the so-called zero-shot learning paradigm. Many algorithms were proposed to solve the bilingual alignment problem in supervised or unsupervised manners. One popular way to extend the bilingual alignment to the multilingual setting is by picking one of the input languages as the pivot language and transiting through that language. However, transiting through a pivot language degrades the quality of translations, since it assumes transitive relations among all pairs of languages. It is often the case that one does not enforce such transitive relations in the training process of bilingual tasks. Therefore, transiting through an uninformed pivot language degrades the quality of translation. Motivated by the observation that using information from other languages during the training process helps improve translating language pairs, we propose a new algorithm for unsupervised multilingual alignment, where we employ the barycenter of all language word embeddings as a new pivot to imply translations. Instead of going through a pivot language, we propose to align languages through their Wasserstein barycenter. Our motivation behind this is that we can encapsulate information from all languages in the barycenter and facilitate bilingual alignment. We evaluate our method on standard benchmarks and demonstrate that our method outperforms state-of-the-art approaches. The barycenter is closely related to the joint mapping for all input languages hence encapsulates all useful information for translation. Finally, we evaluate our method by jointly aligning word vectors in 6 languages and demonstrating noticeable improvement to current state-of-the-art.

Acknowledgements

First of all, I would like to thank my supervisor Yaoliang Yu who supported me throughout my graduate studies at the University of Waterloo. Always being helpful and supportive, Yaoliang guided me in all the time of research and writing this thesis. Yaoliang gave me the freedom to choose my research topic and he also provided lots of technical supports when I'm stuck. I have learned a lot from Yaoliang, not only on how to conduct research but also the way of thinking and solving problems.

A special thanks to Borealis AI, for their physical and technical support. Without their support and funding, this project could not have reached its goal. Cheers to Jakub and Kshitj, researchers from Borealis AI, for stimulating discussions and help writing paper. I am grateful to all those with whom I have had the pleasure to work with at the university and Borealis AI lab.

I also wish to show my gratitude to Jimmy Lin and Pascal Poupart for agreeing to read my thesis. I am gratefully indebted to their very valuable comments on this thesis.

Many of my friends, colleagues, and supportive staff at the university has made my study an enjoyable journey. I wish to thank all the people whose assistance was a milestone in the completion of the thesis.

And finally, to all my family, whose love and guidance are with me in whatever I pursue. Words are not enough to express how thankful I am to their constant support.

Dedication

For my family and friends.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this thesis, we focus on a critical sub-task in machine translation: lexicon translation for multiple languages. The language alignment problem, a long-standing problem in natural language processing (NLP), has received revived interest recently due to new developments in word embeddings and neural approaches. Key inputs for lexicon alignment tasks consists of embedding vectors for each word. Mikolov et al. (2013a) were the first to release their pre-trained model and gave a distributed representation of words. After that, more software for training and using word embeddings emerged.

The rise of continuous word embedding representations has revived research on the bilingual lexicon alignment problem, originally initiated by Rapp (1995) and Fung (1995). Rapp and Fung's initial goal was to learn a small dictionary of a few hundred words by leveraging statistical similarities between two languages. Mikolov et al. (2013a) formulated the problem of aligning word embeddings for two different languages as a quadratic optimization problem with the goal of learning an explicit map between word embeddings. The linear map in Mikolov's formulation aligns language spaces together and enables us to infer meanings for out-of-dictionary words. Zhang et al. (2016), Dinu and Baroni (2015), and Mikolov et al. (2013a) have shown that we can achieve high accuracy for aligning two languages with no parallel data.

Recently, researchers have been interested in extending language alignment to multiple languages with the goal of jointly learning all pairwise translations. One natural approach is to align all languages to a central language. This naturally extends current bilingual alignment methods to the multilingual case while using information from other languages to improve the performance for any language pair. When solving the multilingual alignment problem, Smith et al. (2017) made the assumption that English is a universal language, and treat English as a pivot to transit all languages through. Recently, researchers have explored the possibility to train multiple languages together to achieve better results (Conneau et al., 2017; Taitelbaum et al., 2019a; Nakashole and Flauger, 2017; Alaux et al., 2019).

In this work, we propose a new method for multilingual alignment that uses pre-trained monolingual word embeddings and no other information. Our work uses the barycenter for all languages as the pivot to enforce coherence among language spaces by enabling accurate compositions between language mappings. We show that our approach achieves competitive performance that exceeds the performance of current bilingual alignment methods.

1.1 Background

Many problems in machine learning fall into the category where we have limited training data available to learn a general mapping from one domain to another domain of interest. After training we can then apply the learned mapping to make new predictions. Solutions to such a problem facilitate tasks ranging from machine translation to transfer learning.

One example that falls into this category is the machine translation problem, i.e., aligning words with similar meanings in different languages. While the alignment problem is relatively easy to solve with access to large amounts of parallel data, broader applicability relies on the ability to do so with only monolingual data.

The recent active research on language alignment benefits from the development of word embedding. Word embedding is a significant leap in advancing our ability to analyze words, sentences, and paragraphs. For a long time, words are represented as one-hot vectors. However, such representation makes it impossible for us to explore the similarity between words or predict the context a word may appear in. Bengio et al. (2003) introduced the concept of word embeddings to predict the probability of a word given contexts in a sentence. Word embedding encodes a semantic relationship among words and benefits tackling natural language processing (NLP) problems. Afterward, Collobert et al. (2011) proposed the concept of a pre-trained model, which decoupled the word vector training from downstream training. The proposal made linguistic computation feasible with less computing power. However, this concept did not become popular until the continuous Bag-of-Words model (Mikolov et al., 2013b) and the distributed Skip-gram model (Mikolov et al., 2013a) were developed. These algorithms give rise to the hope that one person can easily find focus words or predict context words with no further information. Mikolov et al. (2013a) were the first to release their pre-trained model and give a distributed representation of words. After that, lots of other software for training and using word embedding emerged, for example, GloVe (Pennington et al., 2014); AllenNLP's Elmo (Gardner et al., 2018); Apache's Deeplearning4j (Team et al., 2016); and Facebook's fastText (Joulin et al., 2016).

Development in computational linguistics and neural language modeling has shown that word embedding can capture the context of a word, containing both semantic and syntactic information (Dinu and Baroni, 2015). This led to the development of the zero-shot learning paradigm as a way to address the manual annotation bottleneck in domains where other vector-based representations must be associated with word labels. It is a fundamental step to make natural language processing more accessible. Building on each other's contributions, NLP researchers can now use pre-trained language models to get those word embedding vectors as the representation for words in machine translation tasks.

In this chapter, we first introduce some preliminary background for language alignment. Then, we formally define the language alignment problem when there are only two languages involved. Finally, we give a review of previous approaches to solving the bilingual alignment problem, as well as current attempts to extend them into the more challenging multilingual setting.

1.2 Language Alignment

Most language models follow the assumption Harris made in 1954: words occurring in similar contexts tend to have similar meanings (Harris, 1954). In the FastText model released by Mikolov et al., researchers discovered an even stronger trait: One can allow languages to be treated like vector spaces with precise mathematical properties. One famous example is the following: King $-$ Man $+$ Woman $=$ Queen.

The rise of continuous word embedding representation has revived researches on the bilingual alignment problem, this time, with a different goal of aligning embedding spaces. The work on aligning bilingual lexicon was started by Rapp (1995) and Fung (1995). Their initial goal was to learn a small dictionary of a few hundred words by leveraging statistical similarities between two languages. We formally define the language alignment problem as follows:

Definition 1 (Bilingual Alignment) *Let $\mathcal{L}_X$ and $\mathcal{L}_Y$ represent two languages, each having n words and m words in their vocabularies, $V_X = \{w_i^X\}_{i=1}^n$ and $V_Y = \{w_j^Y\}_{j=1}^m$, respectively. With pre-trained word embeddings, we have vectors $\mathbf{x}_i \in \mathbb{R}^{d_x}$, $\mathbf{y}_j \in \mathbb{R}^{d_y}$ corresponding to each word w_i^X, w_j^Y in the vocabulary V_X and V_Y, respectively. Let $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top \in \mathbb{R}^{n \times d_x}$ and $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_m]^\top \in \mathbb{R}^{m \times d_y}$. Our goal is to learn an alignment or mapping $T : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$ based on the two sets of word embeddings X and Y. In the following we assume $d_x = d_y$.*

In short, the language alignment problem consists of finding an alignment or mapping $T : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$ given word embeddings of dimensionalities d_1 and d_2 for the vocabularies $V_1 = \{w_i^1\}_{i=1}^n$ and $V_2 = \{w_j^2\}_{j=1}^m$ of languages $\mathcal{L}_1$ and $\mathcal{L}_2$.

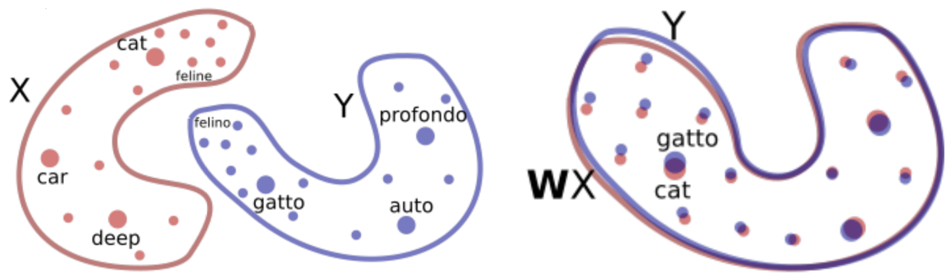

Figure 1.1: Here X denotes the word embedding space for English, where each dot is a word vector lying in that space; Y denotes the embedding space for Italian; and W is the alignment matrix transforming word vectors in X to obtain the smallest distance to vectors in Y.

In one language, words with similar contexts and meanings lie close to each other. So, one natural guess to find the translation for one word in the other language is to find the nearest neighbor in the other language of that word. However, for each language, its word embedding is trained independently on monolingual data, and therefore, lies in different language spaces. Words that are close to each other are not necessarily synonyms.

Figure 1.1 from (Conneau et al., 2017) illustrates one toy example that reveals this issue. Suppose X denotes the embedding space of English, whereas Y is the embedding space of Italian. In this example, *gatto* is the Italian word for cat. However, *gatto* is not the closest word to cat in the space of Y. After aligning the two language spaces, we can see *gatto* is the closest word to cat in the newly transformed space. This toy example illustrates the need to first align the word embedding space for both languages (Language Space Alignment) first. Then, after aligning languages into the same word embedding space, we can align words with similar meanings by finding its closest neighbor in the other language (Translation Inference).

Mikolov et al. (2013c) were the first one observed that there are similarities between monolingual embedding space for different languages even though trained independently. Using a simple linear mapping, Mikolov et al. (2013c) formulated the problem as a quadratic optimization problem learning an explicit map between word embeddings. Later, Xing et al. (2015) showed that the results can be improved by restricting the linear mapping to be orthogonal. They further compared the accuracy on the word translation task using a simple linear mapping against more advanced strategies like multi-layer neural networks. Surprisingly, the linear mapping achieved better results than the more advanced strategy. The rationale of having the orthogonal constraint is that orthogonal maps preserve angles and distances. As per assumption, preserve the distances between words is crucial to solving the word alignment problem.

The bilingual alignment problem was formulated by Mikolov et al. (2013c) and Xing et al. (2015) formulated as the following optimization problem:

Definition 2 (Unsupervised bilingual alignment) *Given two sets of n word vectors $X, Y \in R^{n\times d}$ in language $\mathcal{L}_X$ and $\mathcal{L}_Y$, find orthogonal matrix $Q \in \mathbb{R}^{d\times d}$ and alignment matrix $P \in P_n = B_n \cap \{0,1\}^{n\times n}$ where $B_n = \{P \in R_+^{n\times n}, P1_n = 1_n, P^T 1_n = 1_n\}$ such that*

$$\min_{Q\in\mathbb{R}^{d\times d}:Q^TQ=I} \min_{P\in P_n} \|XQ - PY\|_2^2 \tag{1.1}$$

The problem admits a closed-form solution when we are provided with some aligned data. Introducing supervision into the problem, the problem of aligning language space has the exact form of the Orthogonal Procrustes Problem:

$$\min_{Q\in\mathbb{R}^{d\times d}:Q^\top Q=I} \|XQ - PY\|^2 \tag{1.2}$$

We call the version of the language alignment problem *Supervised Bilingual Alignment*.

The problem was famous in point registration, a fundamental task in computer graphics. When we have a large amount of parallel data during training, we can align the language spaces using Procrustes Matching (Schönemann, 1966): Given Singular Value Decomposition (SVD) $U\Sigma V^T$ of XY^T, the solution to 1.2 is $Q^* = UV^T$ (Schönemann, 1966). Finding such SVD decomposition takes $O(n^3)$. When there's a new word with word embedding vector as v_w in language $\mathcal{L}_X$ that we haven't seen before, we can use the space alignment matrix Q to compute the word vector in the language space of $\mathcal{L}_Y$ by multiplying the vector with the language alignment matrix Q. To get the translation word for v_w, we can find the word vector that is closest to v_wQ in the language space $\mathcal{L}_Y$.

We can solve the supervised version of the language alignment problem efficiently. However, in some applications like in bio-informatics or ancient language translation, we do not have enough parallel data to align language spaces. And that is the setting for unsupervised bilingual alignment, where we need to learn word embedding mapping jointly with the space alignment matrix. Research studies have shown that the need for parallel data supervision can be alleviated with character-level information (Conneau et al., 2017; Alvarez-Melis and Jaakkola, 2018). However, optimization becomes more complicated when we do not have parallel data, so we will leave the current methods for unsupervised bilingual alignment to the next section.

1.3 Unsupervised Bilingual Alignment

In this section, we introduce several methods tackling the unsupervised version of the bilingual alignment problem. There are three main categories for those solutions. First, one can try to relax and optimize the integer linear program in Definition 2 directly using a linear program solver. The advantage is the solution is the most accurate; however, when the input for the problem scale-up, linear program solver quickly becomes computationally inhibit. The second way is to use Block Coordinate Relaxation. In other words, fix each variable and optimize over the other iteratively. This was a popular approach explored by many researchers. However, the state-of-the-art took another direction to solve the problem. We will explain all these different methods in detail.

1.3.1 Direct Optimization

Solving Definition 2 is equivalent to solving the Procrustes matching (PM) problem, which is a fundamental task in computer graphics to align two point clouds. Let $\mathbf{1}_n = [1, \cdots, 1]$ be the vector containing n ones. The PM problem can be formulated as the following linear program:

$$\begin{array}{ll} \text{minimize} & \|QX^T - Y^T P\|_F^2 \\ \text{subject to} & P \in \{0,1\}^{n \times n} \\ & P\mathbf{1}_n = \mathbf{1}_n, P^T\mathbf{1}_n = \mathbf{1}_n \\ & Q \in \mathbb{R}^{d \times d} \\ & Q^T Q = I \end{array} \tag{1.3}$$

The state-of-the-art method to solve 1.3 exactly is maintained by Maron et al. (2016). They relaxed the constraints for PM to be a semi-definite program (SDP-PM). It admits high accuracy at the expense of high computation complexity.

Expanding the constraints in 1.3:

$$\begin{array}{ll} \min\limits_{P,Q} & \|QX^T - Y^T P\|_F^2 \\ \text{subject to} & P\mathbf{1}_n = \mathbf{1}_n, \mathbf{1}_n^T P = \mathbf{1}_n^T \\ & P_j P_j^T = diag(X_j), \quad j = 1 \ldots n \\ & QQ^T = Q^T Q = I \end{array} \tag{1.4}$$

We can see that all polynomials are quadratic in the entries of P and Q, we can define the new variable matrix.

$$Z_j = \begin{bmatrix} P_j \\ [Q] \end{bmatrix} \begin{bmatrix} P_j \\ [Q] \end{bmatrix}^T = \begin{bmatrix} P_j P_j^T & QX_j^T \\ P_j Q^T & QQ^T \end{bmatrix} = \begin{bmatrix} A_j & B_j^T \\ B_j & C \end{bmatrix} \tag{1.5}$$

By expanding 1.4, we rewrite the objective function in terms of Z_j for some constant matrices W_j, since $\|QX^T - Y^TP\|_F^2$ is linear in the entries of Z_j:

$$\|QX^T - Y^TP\|_F^2 = \sum_j \|QX_j^T - Y^TP_j\|_2^2 = \sum_j tr(W_jZ_j) + \text{const} \tag{1.6}$$

The relaxation for the PM problem, PM-SDP can be expressed as

$$\begin{array}{lll} \min\limits_{Z_j,X,R} & \sum_j tr(W_jZ_j) & \\ \text{subject to} & X\mathbf{1} = \mathbf{1}, \mathbf{1}^TX = \mathbf{1}^T & \\ & A_j = diag(X_j) & j = 1...n \\ & tr(H_lC) + b_l = 0 & l = 1...2d^2 \\ & Z_j \succeq \begin{bmatrix} X_j \\ [R] \end{bmatrix} \begin{bmatrix} X_j \\ [R] \end{bmatrix}^T & j = 1...n \end{array} \tag{1.7}$$

The relaxation 1.7 is equivalent to 1.3 by the theorem from (Grone et al., 1984). Furthermore, we can further reduce the size of SDP constraint by setting $X_{ij} = 0$ when we know P_i and P_j should not correspond.

1.3.2 Block Coordinate Relaxation

PM-SDP is good for small size datasets. However, when the size of the dataset exceeds a couple hundred, it becomes computationally unsolvable. So we need a different optimization approach to tackle the problem.

The purpose of using semi-definite linear program is to relax the integer programming problem by removing its integer constraint. We can take advantage the relaxed formulation 1.4 and proceed with different optimization methods. With the same setting as in Definition 2, linear program 1.4 is equivalent as directly removing the integer constraints in the problem:

$$\min_{Q \in O^d} \min_{P \in B_n} \|XQ - PY\|_2^2 \tag{1.8}$$

Even after we remove the integer constraint, we notice that the problem is still non-convex. Neither the constraint sets for P or Q are convex.

For optimization problems like 1.8, a standard way is to iteratively optimizing over each variable with other variables fixed. With the language mapping P fixed, minimizing Q can be solved using Procrustes Analysis, as we discussed in the previous Section 1.2. However, knowing the space alignment matrix, finding the word mapping is not as obvious. We introduce two different ways to do that:

Similarity function

After aligning words of different languages into the same language spaces, words in $\mathcal{L}_X$ to $\mathcal{L}_Y$ are in the same language space so that we can define distances between words with some similarity score function. The similarity function $sim(\cdot,\cdot)$ takes two-word embedding vectors as input and returns a similarity score. We can infer translations of input source word $x \in \mathcal{L}_X$ by finding the word with maximum similarity score $y \in \mathcal{L}_Y$:

$$\hat{y} = \arg\max_{y \in \mathcal{L}_Y} sim(x, y) \tag{1.9}$$

Some common choices for similarity functions are consine function, Euclidean-norms etc. Word embedding representations for languages with similar meanings are close to each other (Bolukbasi et al., 2016). So a naive way is to find the translation for a word is to pick the word vector with maximum similarity with the input word. Let's call the word with maximum similarity nearest neighbor with the input word. In this way, we can imply a permutation matrix indicating translation pairs between $\mathcal{L}_X$ and $\mathcal{L}_Y$.

However, it is generally not true that the nearest neighbor relationship is commutative. If the nearest neighbour for word w^y in language $\mathcal{L}_y$ is w^x in language $\mathcal{L}_x$. The nearest neighbour for w^x in language $\mathcal{L}_y$ might not be w^y. Therefore, picking the nearest neighbor will suffer from the hubness problem: there will be words in one language that are close to many words, so there will be some words that are not similar to any words. Those words are called the "hubs"(Dinu and Baroni, 2015).

One way to correct the hubness problem is to follow the globally-corrected (GC) approach proposed by Dinu (Dinu and Baroni, 2015). To put it in simple words, instead of returning the nearest neighbor of the source word as a solution, it returns the target element y, which has x ranked highest. Let T denote the target space for translations, and $Rank_{y,T}(x)$ denote the rank of an element $x \in T$ w.r.t. its similarity to y and assuming the query space T:

$$GC_1(x, T) = \arg\min_{y \in T} Rank_{y,T}(x) \tag{1.10}$$

The best way to effectively solve the hubness problem is by using the CSLS metric defined as follows

$$CSLS(z, w) = 2\cos(z, w) - \frac{1}{n}\sum_{z' \in N_z(w)} cos(z', w) - \frac{1}{n}\sum_{w' \in N_w(z)} \cos(z, w') \tag{1.11}$$

where $N_z(w)$ is the K-nearest neighbour associate with a word w in the target space, and $N_w(z)$ is the K-nearest neighbour associate with z in the source space. K can be chosen as a hyper-parameter. CSLS expands the space where there is a high density of points, so the hubs become less close to other word vectors than they would otherwise. Conneau et al. (2017) has shown experimentally CSLS achieves a competitive result higher than any other similarity function.

Optimal Transport (OT)

Another way to find translation mapping is by solving the Optimal Transport Problem transporting the language distribution for $\mathcal{L}_X$ to language distribution for $\mathcal{L}_Y$ (Alvarez-Melis and Jaakkola, 2018).

The languages distribution for $\mathcal{L}_X$ and $\mathcal{L}_Y$ are both distributions over their vocabulary word embeddings:

$$\mu_X = \sum_{i=1}^{n} p_i \delta_{x^{(i)}} \quad \mu_Y = \sum_{j=1}^{m} q_j \delta_{y^{(i)}} \tag{1.12}$$

where p_i is the probability word w_i^x occurs in language $\mathcal{L}_X$, usually frequency of word w_i^x occurring in its training documents; Similarly, q_j is the probability word w_j^y occurs in language $\mathcal{L}_Y$. and δ is the one-hot function

$$\delta_a(x) = \begin{cases} 1, & \text{if } x = a \\ 0, & \text{otherwise} \end{cases} \tag{1.13}$$

In the following sections, we call p, q the probability vectors for languages.

With the probability representation of languages, it is natural to adopt optimal transport to find a mapping from $\mathcal{L}_X$ to $\mathcal{L}_Y$. For some cost function $c(x, y)$ defining the effort of changing from x to y. The correspondence between words in $\mathcal{L}_X$ and words in $\mathcal{L}_Y$ is captured in the transportation map T realizing

$$\inf_T W_p(\mu, \nu) = \inf_T \left\{ \int_X c(x, T(x)) d\mu_X(x) \quad \middle| \quad T_\# \mu_X = \mu_Y \right\}^{1/p} \tag{1.14}$$

The equation 1.14 is the definition for p-Wasserstein between distributions μ and ν.w

Here, $\#$ is the push forward operator defined as follows:

Definition 3 *For a continuous map $T : X \to Y$, the push-forward operator $T_\# : M(x) \to M(Y)$ pushes one distribution to another. For discrete measures, push forward operation operates on support of the measure, updating positions of all points to $T(x_i)$, without changing its probability weight.*

Formally, let α be any discrete probability measure having weight a_i at location x_i, the push forward measure is the probability

$$\alpha = \sum_i a_i \delta_{x_i} \quad \xrightarrow{\textit{push forward}} \quad T_{\#}\alpha = \sum_i a_i \delta_{T(x_i)} \tag{1.15}$$

When α is a continuous random measure in space χ, the push forward $\beta = T_{\#}\alpha \in M(\gamma)$ is a random measure on space γ. For any measurable set $B \subset \gamma$, one has

$$\beta(B) = \alpha(\{x \in \chi : T(x) \in B\}) = \alpha(T^{-1}(B)) \tag{1.16}$$

In the language alignment problem, the language distributions are discrete, because we have finite words in a dictionary. The Optimal Transport(OT) plan between two discrete distributions can be seen as a probability coupling $\Gamma \in R_+^{n\times m}$, where $\Gamma_{i,j}$ denote similarity between w_i^x and w_j^y. The coupling can be realized by minimizing the loss function:

$$\min_{\Gamma \in \Pi(p,q)} < \Gamma, C > \tag{1.17}$$

where C is the cost matrix containing the cost of transporting words in $\mathcal{L}_X$ to $\mathcal{L}_Y$, i.e $C_{ij} = dist(w_i^x, w_j^y)$ for some dist$(\cdot,\cdot)$; and Γ is the transport plan in the polytype $\Gamma(p,q) = \{\Gamma \in R_+^{n\times m} | \Gamma 1_n = p, \Gamma^T 1_n = q\}$

The entry at row i and column j in the optimal OT transport plan minimizing 1.17 can be seen as the probability word w_i^x map to w_j^y. The method is simple, efficient, and theoretically motivated. The advantage of using Wasserstein distance is that optimal transport gives a natural geometry for probability measures supported on geometric space. When two words are close to each other, we put more weight on them, mapping to each other to minimize the loss. Solving the language alignment problem also benefit from the recent advance in optimal transport distance computation: smooth the OT problem with a negative entropy, one can compute the optimum efficiently with the Sinkhorn algorithm (Cuturi, 2013).

1.3.3 State-of-the-art

Alternatively optimizing over each variable gives a moderate result at the expense of intensive computation.

The fundamental reason behind the need for iterative minimization is that classic OT requires a distance between vectors from two different domains. Such a metric isn't always possible when two spaces have different dimensions. However, the rise of Gromov-Wasserstein distance gives hope for us to bypass the fuss of computing such alignment before computing translations. The difference between Wasserstein distance and Gromov-Wasserstein distance is that classic Wasserstein distance requires a distance between vectors across two domains; Whereas Gromov-Wasserstein distance operates on distances between pairs of points calculated within each domain and measures how these distances compare to those in the other domain.

Recently applied to the problem of cross-lingual word embedding alignment, Alvarez-Melis and Jaakkola (2018) have shown that pure Gromov-Wasserstein approach leads to high-quality solutions in practice (Vayer et al., 2018). It is currently the state-of-the-art method for the bilingual alignment problem.

On each language, define a loss function on pairs of words: $L : R \times R \to R$. Distance between all pairs of words gives the within-domain similarity matrix C where $C_{ij} = c(w_i, w_j)$. In two different languages, calculating within-domain similarity matrices and frequency vector for each language and get (C, p) and (C', q).

Expressing the discrepancy between distances $c(x^{(i)}, x^{(k)})$ and $c'(y^{(j)}, y^{(l)})$ as a 4-th order tensor $L \in R^{n \times n \times m \times m}$, where $X_{ijkl} = L(C_{ik}, C'_{jl})$

The Gromov-Wasserstein problem is defined as

$$GW(C, C', p, q) = \min_{\Gamma \in \Pi(p,q)} \mathcal{E}_{C,C'}(T) \tag{1.18}$$

where

$$\mathcal{E}_{C,C'}(T) =< L(C, C') \otimes T, T >= \sum_{i,j,k,l} L(C_{i,k}, C'_{j,l}) \Gamma_{ij} \Gamma_{kl} \tag{1.19}$$

The transportation coupling Γ_{ij} denote the likelihood two words are translations of each other. Only when both $(w_i^x,\, w_j^y)$ and $(w_k^x,\, w_l^y)$ are translation pairs, will we add X_{ijkl} into the total loss in 1.19.

The problem is substantially harder than the original optimal transport problem 1.14, since the second-order equation is non-linear and non-convex. If one were to solve the problem naively with first-order methods, the complexity of solving is $O(N_1^2 N_2^2)$

With some suitable choice of the distance metric L, Peyre has shown that solving 1.18 with projected gradient descent yields an algorithm with $O(N_1^2 N_2 + N_1 N_2^2)$ cost (Peyré et al., 2016):

Theorem 1.3.1 *(Peyré et al., 2016) If the loss can be expressed as*

$$L(a, b) = f_1(a) + f_2(b) - h_1(a)h_2(b) \tag{1.20}$$

for functions (f_1, f_2, h_1, h_2)*, then for any* $\Gamma \in \Pi(p, q)$*,*

$$L(C, C') \otimes T = c_{C,C'} - h_1(C)\Gamma h_2(C')^T \tag{1.21}$$

where $c_{C,C'} = f_1(C)p1_m^T + 1_n q^T f_2(C')^T$

There are two special cases the author gave explicit decomposition (f_1, f_2, h_1, h_2) for: For the case $L = L_2$, 1.3.1 for $f_1(a) = a^2$, $f_2(b) = b^2$, $h_1(a) = a$ and $h_2(b) = 2b$. And for the KL loss satisfy 1.3.1 for $f_1(a) = alog(a) - a$, $f_2(b) = b$, $h_1(a) = a$ and $h_2(b) = log(b)$.

Consider the entropic approximation of GW formulation 1.3.1

$$GW_\epsilon(C, C', p, q) = \min_{\Gamma \in \Pi(p,q)} \mathcal{E}_{C,C'}(\Gamma) - \epsilon H(T) \tag{1.22}$$

After adding an entropic constraint, we can use the projected gradient descent to solve the non-convex optimization problem.

Iterations for this algorithm is given by $T \leftarrow \text{Proj}_{C_{p,q}}^{KL}(T \odot e^{-\tau(\nabla \mathcal{E}_{C,\overline{C}}(T) - \epsilon \nabla H(T))})$ with step size τ, and KL projector for any matrix K defined by $\text{Proj}_{C_{p,q}}^{\text{KL}}(K) = \arg\min_{T' \in C_{p,q}} KL(T'|K)$. When we choose step size $\tau = 1/\epsilon$, the iteration becomes $T \leftarrow \tau(L(C, C') \otimes \Gamma, p, q)$. So we can rewrite the iteration to be a fix-point iteration in which each update of T involves a Sinkhorn projection.

The Gromov-Wasserstein approach directly compares the distances between pairs of words calculated in their own language spaces. It is a simple, one-step optimization requires none supervision. However, the algorithm is only suitable to small-to-medium size vocabulary, since optimization will become prohibitive for large size problems.

To scale up the algorithm for larger size problems, Melis proposes to use the GW approach to map a small subset of the vocabulary and learn the orthogonal mapping of language spaces with Procrustes Matching. Then use the iterative two-step optimization procedure similar to Conneau et al. (2017).

1.4 Multilingual Alignment

In Section 1.2, we pointed out the general direction for aligning lexicons from two different languages. Afterwards, in section 1.3, we gave a walk-through of different methods to do the bilingual alignment in unsupervised fashion. In this section, we are going to introduce our main problem: how to align multiple languages in an unsupervised way.

To setup the stage, let $\{\mathcal{L}_i\}_{i=1}^m$ be m languages, each of which is represented by a vocabulary $\mathcal{V}_i$ consisting of n_i respective words.

Following Mikolov et al. (2013a) we assume a monolingual word embedding $X_i = [\mathbf{x}_{i,1}, \cdots, \mathbf{x}_{i,n_i}]^T \in \mathcal{R}^{n_i \times d_i}$ for each language $\mathcal{L}_i$ has been trained *independently* on its own data. We are interested in finding *all* pairwise mappings $T_{i \to k} : \mathcal{R}^{d_i} \to \mathcal{R}^{d_k}$ that translate a word $\mathbf{x}_{i,j_i}$ in language $\mathcal{L}_i$ to a corresponding word $\mathbf{x}_{k,j_k} = T_{i \to k}(\mathbf{x}_{i,j_i})$ in language $\mathcal{L}_k$. In the following, for the ease of notation, we assume without loss of generality that $n_i \equiv n$ and $d_i \equiv d$. Note that we do not have access to any parallel data, i.e., we are in the much more challenging unsupervised learning regime.

Definition 4 (Multilingual Alignment) *Given m input languages, $\{\mathcal{L}_i\}_{i=1}^m$, each of which is represented by a vocabulary $\mathcal{V}_i$ consisting of n_i respective words. With pre-trained word embeddings, we have monolingual word embedding $X_i = [\mathbf{x}_{i,1}, \cdots, \mathbf{x}_{i,n_i}]^T \in \mathcal{R}^{n_i \times d_i}$ for each language $\mathcal{L}_i$. Our goal is to learn an alignment or mapping for all languages $T_{i \to k} : \mathcal{R}^{d_i} \to \mathcal{R}^{d_k}$.*

A simple way to generalize bilingual methods into the multilingual case is by choosing one language as a pivot language and infer translations for language pairs by transiting through the pivot language (Smith et al., 2017). Assuming English is the pivot language, we only need to find language translation pairs involving English. In other words, finding permutation matrices $\{P_i\}_{i=2}^m$ aligning word vectors and orthogonal matrices $\{Q_i\}_{i=2}^m$ mapping language $\mathcal{L}_i$ into the language space of English. The objective function to simultaneously aligning all languages to English looks like:

$$\min_{Q_i \in O_d, P_i \in P_n} \sum_i l(X_i Q_i, P_i X_0) \tag{1.23}$$

This approach is a simple extension for current bilingual alignment methods. However, the disadvantage is that the accuracy of indirect word translations between pairs of languages that do not include the pivot is not guaranteed. Using English as a pivot language degenerate the quality of indirect word translation obtained.

One possible reason for degrade for translation quality might be incoherence in language mappings. Nakashole and Flauger (2017) showed that constraining coherent word alignments between triplets of nearby languages improves the quality of induced bilingual lexicons. Lot of researchers took the approach of aligning all languages into a coherence space and train languages jointly (Taitelbaum et al., 2019b; Chen and Cardie, 2018; Alaux et al., 2019).

Taitelbaum et al. (2019b) were one of the researchers motivated by the fact that borrowing information from other languages during the training process will improve bilingual translation tasks. They extended Procrustes Matching to the multilingual case learning all mappings jointly. Associating each language Li with an orthogonal matrix Q_i, we transform all languages into a shared potential space, which might be different from all language spaces.

In the bilingual case, given two language word embedding matrices X and Y, the Procrustes Matching problem try to minimize on the loss:

$$S(Q) = \min_{Q} \|XQ - Y\|_2^2 = \min_{Q} \sum_{t=1}^{n} \|Qx_t - y_t\|^2 \tag{1.24}$$

where x_t is the word embedding vector representing the t^{th} word in language $\mathcal{L}_X$.

In the multilingual case, Taitelbaum et al. (2019b) proposed an extension of Procrustes Matching: Assuming we have paralleled data (which might be results of last iteration training) across all languages. Let n_{ij} be the size of dictionary for language $\mathcal{L}_i$ and language $\mathcal{L}_j$. And let $(p_{ij,t}, p_{ji,t})_{t=1}^{n_{ij}}$ be translation pairs for language i and language j language.

Minimizing the following mean-square error, we can find orthogonal matrices Q_i mapping corresponding words from different languages into the shared vector space [1]:

$$S(Q_1, \cdots, Q_k) = \sum_{i<j} \sum_{t=1}^{n_{ij}} \|Q_i p_{ij,t} - Q_j p_{ji,t}\|^2 \tag{1.25}$$

One thing to notice is that there's no closed form for equation 1.25. In their paper, Taitelbaum et al. (2019b) proposed an efficient algorithm to optimize the cost. They decouple all orthogonal matrices by computing the representation of $p_{ji,t}$ in the underlying common space: $y_{j,t} = T_j p_{ji,t}$. After the decoupling, one can optimize the score 1.26 with respect to each mapping matrix Q_i.

Rewrite the objective score in 1.25 to be a loss function depending on one variable Q_i:

$$S(Q_i) = \sum_{j \neq i} \sum_{t=1}^{n_{ij}} \|Q_i p_{ij,t} - y_{j,t}\|^2 + \text{const} \tag{1.26}$$

where $y_{j,t} = Q_j p_{ji,t}$ is the representation of $p_{ji,t}$ in the underlying space, calculated by

$$y_t = \sum_{j \neq i} Q_j p_{ji,t} p_{ij,t}^T \tag{1.27}$$

Equation 1.26 is in the form of Orthogonal Procrustes Problem again, so $Q_i = UV^T$ where $U\Sigma V^T$ is the SVD of

[1]The notation $(p_{ij,t}, p_{ji,t})$ means the t^{th} corresponding word in the dictionary in the i and j language

$$M_i = \sum_{j \neq i} \sum_t Q_j p_{ji,t} p_{ij,t}^T = \sum_{j \neq i} Q_j \underbrace{\sum_t p_{ji,t} p_{ij,t}^T}_{\text{precompute with } C_{ij} = C_{ji}^T = \sum_t p_{ji,t} p_{ij,t}^T} \tag{1.28}$$

Putting all the above together, Taitelbaum et al. (2019b) proposed the Multi Pairwise Procrustes Analysis (MPPA) algorithm. With the algorithm, the total loss for 1.25 decreases monotonically and converges to a local minimum point.

The multilingual alignment problem has drawn more and more attention lately. Besides Smith et al. (2017) and Taitelbaum et al. (2019b), Chen and Cardie (2018) also tried to extend the work of Conneau et al. (2017) into multilingual case using adversarial algorithms.

Our work is largely motivated by the current state-of-the-art framework for unsupervised multilingual alignment proposed by Alaux et al. (2019), which we review below first. Along the way we point out some crucial observations that motivated our further development.

For each language $\mathcal{L}_i$, Alaux et al. (2019) associate it with space alignment matrix Q_i aligning languages to the shared space, and language alignment matrices to all other languages $P_j, \forall j \neq i$.

The objective function to achieve such target is:

$$\min_{Q_i \in O_d, P_{ij} \in P_n} \sum_{ij} \alpha_{ij} l(X_i Q_i, P_{ij} X_j Q_j) \tag{1.29}$$

where $Q_i \in \mathcal{O}_d$ is a $d \times d$ orthogonal matrix and $P_{ik} \in \mathcal{P}_n$ is an $n \times n$ permutation matrix.

Notice that one way to enforce compositionality is by adding constraint on mapping matrices to enforce coherence, i.e: $P_{ij} = P_{ik} * P_{kj}, \forall k \neq i$ and $k \neq j$. However, this will lead to an estimation of $O(N^2)$ mappings. Instead, the framework leverage the fact that all vector are mapped to a common space, and that enforced good alignments within the space. So Alaux et al. (2019) introduced the weight $\alpha_{ij} > 0$ represents the importance of alignment quality between language i and language j. Those weights can be chosen to reflect prior knowledge about relations between languages.

Since Q_i is orthogonal, this approach ensures transitivity among word embeddings: Q_i maps the i-th word embedding space X_i into a common space X, and conversely $Q_i^{-1} = Q_i^T$ maps X back to X_i. Thus, $Q_i {Q_k}^T$ maps X_i to X_k, and if we transit through an intermediate word embedding space X_t, we still have the desired transitive property $Q_i Q_t^T \cdot Q_t Q_k^T = Q_i Q_k^T$.

One caveat is that the approach took by Smith et al. (2017) falls into the framework when the loss function is Euclidean loss, and potential shared space is the language space of English.

When using the Euclidean distance as loss function, the formulation 1.29 becomes:

$$\min_{Q_i \in \mathcal{O}_d, P_i \in \mathcal{P}_n} \sum_{i,k} \|P_i X_i Q_i - P_k X_k Q_k\|^2 \tag{1.30}$$

which is easily seen to be equivalent to

$$\min_{Q_i \in \mathcal{O}_d, P_i \in \mathcal{P}_n} \min_{X \in \mathbb{R}^{n \times d}} \sum_{i} \|P_i X_i Q_i - X\|^2, \tag{1.31}$$

where X clearly admits the closed-form solution:

$$X = \bar{X} := \frac{1}{m} \sum_{i=1}^{m} P_i X_i Q_i. \tag{1.32}$$

Thus, had we known the "mean" language $\bar{X}$ beforehand, the joint alignment approach of Alaux et al. (2019) would reduce to a separate alignment of each language X_i to the *'mean'* language $\bar{X}$ that serves as the pivot. An efficient optimization strategy would then consist of alternating between separate alignment (i.e., computing Q_i and P_i) and computing the pivot language (i.e., (1.32)).

The discovery connects with the approach took by Taitelbaum et al. (2019a), whom designed a better representation for words using auxiliary translation information from other languages. After aligning all languages to a potential shared space by associating each language $\mathcal{L}_i$ with an orthogonal matrix Q_i, and an alignment matrix P_i, Taitelbaum et al. (2019a) introduced the concept of an auxiliary translation.

Standard ways to imply assignment matrices are introduced in Section 1.3.2. At the inference step, we pick the word vector with maximum similarity score to the input word vector at the inference step. Common choices for similarity functions are: $sim(x, y) = cos(Q_i x, Q_j y)$, Inverted soft max (ISF) (Smith et al., 2017), and Cross-domain Similarity Local Scaling (CSLS) (Conneau et al., 2017).

Current word translation methods take into account only the source and target languages. Taitelbaum et al. (2019a) designed a better representation for the source word in the shared space to imply translations in other languages. For each word $x \in \mathcal{L}_i$ to be translated to language $\mathcal{L}_j$,Taitelbaum et al. (2019a) defined its auxiliary translation using information from all other languages but L_j:

$$z = \frac{1}{k-1} \sum_{m \neq j} Q_m y_m \tag{1.33}$$

where

$$y_m = \arg\max_{y \in L_m} CSLS(Q_i x, Q_m y) \tag{1.34}$$

The closest word to x in language $\mathcal{L}_m$ is y_m, so we use y_m as the auxiliary information for word x in $\mathcal{L}_m$. The auxiliary translation in Equation 1.33 coincide with our "mean" language built in Equation 1.32.

Besides a simple average, Taitelbaum et al. (2019a) also introduce several other ways to build auxiliary translations:

NT (Nearest Translation) Averaging the source word with auxiliary translation in all languages but L_j, we build a better representation for x. To infer its translation in language j, we find the closest word in L_j:

$$\hat{y} = \arg\max_{y \in V_j} CSLS(z, Q_j y) \tag{1.35}$$

The NT approach directly uses the representation in 1.33 to find a nearest neighbour target space L_j.

The average across all auxiliary language may hurt the performance of some language pairs. Generally, translations to auxiliary languages can yield words that are far from the source word in the shared space, and therefore, lead to incorrect results.

CNT (Conditional Nearest Translation A simple way to mitigate the problem is by putting an "if" statement into the average. If the auxiliary information in one language is too far away from x (the original source word), then we do not include the auxiliary information in that language into the average for 1.33. In other words, a language L_m is selected as auxiliary language only if the translated word y_m is closer than target word y_j to the source word x: $CSLS(Q_i x, Q_m y_m) > CSLS(Q_i x, Q_j y_j)$

CAT (Conditional All Translations Another way to mitigate the problem is by putting a weight on auxiliary information. The weight Taitelbaum proposed is $\frac{1}{2}$ on x, and even weight on auxiliary information. Let $S = \{m \| CSLS(Q_i x, Q_m y_m) > CSLS(Q_j y_j) \forall m \neq i$ and $m \neq j\}$

$$z \leftarrow \frac{1}{2} Q_i x + \frac{1}{2|S|} \sum_{m \in S} Q_m y_m \tag{1.36}$$

These are the three different ways to build the augmented representation for the source word. Among them, CAT especially improves word translations, showing 1.7% improvement for all translation pairs on average (Taitelbaum et al., 2019a).

As we reviewed in this section, the most advanced multilingual algorithms we reviewed are projecting all languages into a common space and build a "mean" languages to help inference.

However, there are some problems with the current formulation in Equation 1.29: First, a permutation assignment is a 1-1 correspondence that completely ignores polymorphism in natural languages, that is, a word in language $\mathcal{L}_i$ can correspond to multiple words in language $\mathcal{L}_k$. To address this, we propose to relax the permutation P_i into a coupling matrix that allows splitting a word into different words. Second, the pivot language in (1.32), being a simple arithmetic average, may be statistically very different from any of the m given languages. Besides, intuitively it is perhaps more reasonable to allow the pivot language to have a larger dictionary so that it can capture all linguistic regularities in all m languages. To address this, we propose to use the Wasserstein barycenter as the pivot language.

Chapter 2

Barycenter

In Chapter 1, we defined the problem we are trying to solve in this thesis: the Unsupervised Multilingual Alignment problem. We analyzed the shortcome of current methods and proposed to use the Wasserstein barycenter as a pivot. We are going to propose our algorithms in Chapter 3. So in this chapter, we are going to explain the preliminaries: definition of the barycenter and efficient algorithms to compute barycenters.

2.1 Barycenter Definition and Properties

Barycenter has been a fascinating topic in many areas: In physics, it denotes the center of mass; In geometry, it is a synonym for centroid; In astrophysics, barycenter denotes the center of mass of two or more bodies orbits each other. In mathematics, barycenter for input distributions on given weights was defined by Agueh and Carlier. It naturally generalizes McCann's interpolation in the case of more than two distributions (Agueh and Carlier, 2011).

Definition 5 (Wasserstein Barycenter) *For probability measures $\nu_1, \cdots, \nu_n$ defined on space $\mathcal{X}$, and positive barycentric coordinates $\lambda_1, \cdots, \lambda_n$ summing to 1, we define the barycenter of $\nu_1, \cdots, \nu_n$ to be the solution of the minimization problem*

$$\inf_{\mu} \sum_{i=1}^{n} \lambda_i W_2^2(\nu_i, \mu) \tag{2.1}$$

where $W_2^2(\nu_i, \mu)$ denote the two-Wasserstein distance defined in 1.14 when $p = 2$.

The Wasserstein Barycenter can be seen as a Fréchet mean of distributions over the Wasserstein metric. So it generalizes the problem of computing the mean in unsupervised learning. One can see the k-means problem as a variational barycenter problem with constraint: when the input probability measures $\nu_1, \cdots, \nu_n$ are supported on a discrete finite subset of $\mathcal{X} = R^d$, and the cost is squared Euclidean distance, constraining μ to be a discrete measure with k support in the barycenter problem 2.1 is equivalent to the usual k-means problem.

Besides proposing the definition of barycenter, Agueh and Carlier also studied properties of barycenters, established existence and uniqueness condition for solutions to 2.1. One thing that makes barycenter particularly interesting is that it is related to the quadratic multi-marginal optimal transport problem considered by Gangbo and Świech (1998). This means if we can solve the multi-marginal problem, we also solve the barycenter problem.

The multimarginal optimal transport problem is defined as following:

Definition 6 (Multi-Marginal Optimal Transport Problem) *For every $x = (x_1, \cdots, x_p) \in (R^d)^p$, we define*

$$T(x) = \sum_{i=1}^{p} \lambda_i x_i \tag{2.2}$$

The multimarginal optimal transport problem seeks a probability measure on $(R^d)^p$ having $\nu_1, \cdots, \nu_p$ as marginals, minimizing

$$\inf \left\{ \int_{(R^d)^p} (\sum_{i=1}^{p} \frac{\lambda_i}{2} |x_i - T(x)|^2) d\gamma(x_1, \cdots, x_p), \gamma \in \Pi(\nu_1, \cdots, \nu_p) \right\} \tag{2.3}$$

The multi-marginal OT problem has a unique solution when all distributions are continuous. The solution to multi-marginal OT problem 2.3 is the Gangbo-Święch map constructed by Gangbo and Święch (Gangbo and Świech, 1998). The map γ is constructed as following: $\gamma = (T_1^1, \cdots, T_p^1)\#\nu_1$ with $T_i^j = \nabla u_i^* \circ \nabla u_j$ being Gangbo-Święch maps between reference measures v_j and v_i where u_i are strictly convex potentials defined by

$$u_i(x) = \frac{\lambda}{2}|x|^2 + \frac{g_i(x)}{\lambda_i} \tag{2.4}$$

and $(g_1, \cdots, g_p)$ are convex potentials solves the dual of 2.3

$$\inf \left\{ \sum_{i=1}^{p} \int_{R^d} g_i d\nu_i, \sum_{i=1}^{p} g_i(x_i) \geq \sum_{1 \leq i < j \leq p} \lambda_i \lambda_j x_i x_j, \forall x \in (R^d)^p \right\} \tag{2.5}$$

The relationship between multimarginal OT problem 6 and Barycenter problem 5 is capsuled in the following theorem by Agueh and Carlier (Agueh and Carlier, 2011):

Theorem 2.1.1 *The multimarginal optimal transport problem reduces to the barycenter problem. The solution of 2.1 is given by $\mu = T\#\gamma$ where T is defined by 2.2 and γ is the solution of 2.3.*

From the relationship to multi-marginal optimal transport problem, one can induce the following characteristic of barycenter:

$$v = \big(\sum_{i=1}^{p} \lambda_i T_i^1\big)\#\nu_1 = \big(\sum_{i=1}^{p} \lambda_i T_i^j\big)\#\nu_j \tag{2.6}$$

However, although theorem 2.1.1 is a beautiful theorem, it is not helpful when it comes to solving the barycenter. One of the conditions for the uniqueness of the solution to 2.3 is that all input distributions have to be continuous. However, in almost all applications, input distributions are observation from real life, so naturally, they are discrete distributions. The Gangbo-Święch map is only a theoretical formulation for the joint map. We lose the uniqueness property for the solution to the multi-marginal mapping problem.

2.2 Computation of barycenter

In theory, the existence of barycenter is shown by constructing Gangbo-Swiech map and apply 2.1.1 to derive the barycenter. However, in practice, such maps are computationally infeasible to compute. So to use barycenter for various tasks in computer vision, we need more practical ways to compute the barycenter.

In simple cases, when the optimal transport marginals have closed-form distribution, for example, elliptically contoured distributions, one can directly work with distribution and algebraically find the representation of barycenter. Those cases happen when we can cast the Wasserstein distance as a simpler distance between suitable representations of these distributions. However, in most applications, instead of a continuous model with closed-form distance, one need to resort to a careful discretization. There are two categories for discretization methods: Eulerian and Lagrangian (Peyré, Cuturi, et al., 2019). The difference between Eulerian and Lagrangian is that in Lagrangian representation $\frac{1}{n}\sum_i \delta_{x_i}$, the distribution has x_i's as its support locations; whereas in Eulerian representation $\sum_i a_i\delta_{x_i}$, each x_i represent a location in the whole space, and a_i is the weight on it. For the language model, the Lagrangian discretization method is more suitable, since all word embedding vectors are lying in high dimensional spaces. So we want to compute the barycenter with weights $\{a_i\}_{i=1}^n$ on support locations x_i's. However, in the chapter, we are going to discuss more general cases to compute the barycenter. So in section 2.4, we discuss method to solve for the Lagrangian case. In 2.3, we solve the problem in Eulerian case.

2.3 Fix support barycenter

When measures are supported on low dimensional space or intrinsically discrete problems, one takes the Eulerian discretization. When applied to problems where observations can take continuous values in high-dimensional spaces, a Lagrangian perspective is usually a more suitable choice. The difference between the two approaches is that Eulerian suppose all distributions α in optimal transport problem to be supported on fixed n locations $\{\delta_{x_i}\}_{i=1}^n$. The parameterized measure α is entirely represented through the weight vector $a : \theta \to a(\theta) \in \Sigma_n$; whereas Lagrangian allows free support locations $\{\delta_{x(\theta)_i}\}_{i=1}^n$.

In this section, we assume input distributions are Eulerian discretized. In the following section, we discuss the Lagrangian case.

Given probability measures $\nu_1 = (C_n, p_1), \cdots, \nu_n = (C_n, p_n)$ defined on space $\mathcal{X}$, and barycenter coordinates $\lambda_1, \cdots, \lambda_n$ summing to 1, the fix-support barycenter problem try to find the best weight vectors $a = \{\}_{i=1}^m$ for fixed support locations $X = \{x_i\}_{i=1}^m$.

$$\inf_a \sum_{i=1} \lambda_i W_2^2(\nu_i, X) \tag{2.7}$$

One thing to notice is that the problem can be rewritten as a linear equation and use standard optimization methods to optimize it. This was the solution Cuturi and Doucet exploited in 2013 (Cuturi and Doucet, 2014). Benamou et al. also explored the approach of using iterative Bregman projection to derive an iterative algorithm for fast computation of the barycenter (Jean-David Benamou and Guillaume Carlier and Marco Cuturi and Luca Nenna and Gabriel Peyré, 2015). Besides devising algorithms that converge to the true barycenter, researchers also try to approximate the barycenter with other definitions, such as sliced Wasserstein barycenter (Rabin et al., 2011), convolutional Wasserstein barycenter (Solomon et al., 2015), etc. The following subsections will discuss those different approaches in detail.

2.3.1 Sliced-Wasserstein Distance

The Wasserstein distance $W(\mu, \nu)$ between two discrete measures $\mu = \{x_i\}_{i \in I} \subset R^d$ and $\nu = \{y_i\}_{i \in I} \subset \mathcal{R}^d$ of the same size $|I| = N$ can be expressed as

$$W(\nu,\mu)^2 = \min_{P\in\mathcal{P}_N} \sum_{i,j\in I^2} P_{i,j}\|x_i - y_j\|^2 \tag{2.8}$$

In general, the Wasserstein distance 2.8 between two discrete measures can be solved in $O(N^{2.5}\log(N)$ operations with standard linear programming algorithms. This could be computationally too demanding for tasks like image processing, where N could be quite large.

However, the barycenter problem is much easier to solve if all inputs are in one dimension. By sorting input points to be in ascending order and take ordered pairs as coupling, we can obtain an optimal coupling in $O(N\log(N))$ operations.

Let σ_X and σ_Y be permutations that sort points in X in ascending sequence:

$$\forall 0 \le i < N-1, \quad X_{\sigma_X(i+1)} \quad \text{and} Y_{\Sigma_Y(i)} \le Y_{\Sigma_Y(i+1)} \tag{2.9}$$

Then the optimal permutation σ^* that minimizes 2.8 is

$$\sigma^* = \sigma_Y \odot \sigma_X^{-1} \tag{2.10}$$

The bottleneck of constructing such an algorithm lies in sorting points in X and Y so that the assignment could be computed in $O(N\log(N))$.

Rabin et al. (2011) took advantage of the computation in 1D and proposed to project the high dimensional input onto a 1-dimensional space. Their main idea is to project Wasserstein distance between distributions in high dimensional space to one dimension subspace, and approximate the Wasserstein distance with a series of 1D optimal assignments.

They define an approximation of 2.8: sliced Wasserstein distance $\tilde{W}(\mu,\nu)$

$$\begin{gathered}\tilde{W}(\mu,\nu)^2 = \int_{\theta\in\Omega} W(\mu_\theta,\nu_\theta)^2 d\theta \\ \texttt{where } \mu_\theta = \{<\mu_i,\theta>\} \subset R \texttt{ and } \Omega = \{\theta \in R^d/|\theta| = 1\}\end{gathered} \tag{2.11}$$

Generalizing this to the definition of barycenter, Rabin et al. (2011) stated that one way to approximate the barycenter is through the sliced Wasserstein barycenter which minimizes the following loss function $\tilde{E}(X)$:

$$\tilde{E}(X) = \inf \sum_{i=1}^{p} \lambda_i \tilde{W}(\nu_i,\mu)^2 = \int_{\theta\in\Omega} \sum_{i=1}^{p} \lambda_i W(\nu_{i_\theta},\mu_\theta)^2 d\theta \tag{2.12}$$

Computing the descent direction with the whole set of directions $\theta \in \Omega$ is too expensive, so Rabin proposed a stochastic descent algorithm.

The descent update reads:

$$\mu^{(k+1)} = \mu^{(k)} - \eta_k H_k^{+} \sum_{\theta \in \Omega_k} \nabla \tilde{E}_\theta(\mu^{(k)}) \tag{2.13}$$

where the hessian matrix read

$$H_k^{+} = \sum_{\theta \in \Omega_k} \theta \theta^T \tag{2.14}$$

$$\nabla \tilde{E}_\theta(X^{(k)})_i = H_k X_i^{(k)} \sim \sum_{\theta \in \Omega_k} \sum_{i=1}^{p} \lambda_i < \nu^j_{\sigma_{\theta^j(i)}}, \theta > \theta \tag{2.15}$$

This algorithm is effective in lower dimensions. However, it may not be able to generalize to non-Euclidean spaces or work for $d \geq 4$.

Recently, Wasserstein distance has drawn more attention, as it has many applications on image processing, robust machine learning, etc. All those applications involve high dimensional inputs, so one needs more robust algorithms. In the next subsections, we will discuss other methods to approximate barycenter.

2.3.2 Bregman projection for Wasserstein distance

Another relaxation for problem 2.1 is to add an entropy regularizer (Jean-David Benamou and Guillaume Carlier and Marco Cuturi and Luca Nenna and Gabriel Peyré, 2015; Cuturi and Doucet, 2014): The entropy regulated version of Wasserstein 2 distance between μ_0 and μ_1:

$$W^2_{2,\epsilon}(\mu_0, \mu_1) = \inf_{\pi \in \Pi} \left[\int \int_{M \times M} d(x,y)^2 \pi(x,y) dxdy - \epsilon H(\pi) \right] \tag{2.16}$$

with entropy regularizer

$$H(\pi) = - \int \int_{M \times M} \pi(x,y) \ln \pi(x,y) dxdy \tag{2.17}$$

The Kullback-Leibler (KL) divergence between absolutely continuous meansure $\pi \in Prob(M \times M)$ and positive function $\mathcal{K}$ on $M \times M$ as

$$KL(\pi | \mathcal{K}) = \int \int_{M \times M} \pi(x,y) \left[\ln \frac{\pi(x,y)}{\mathcal{K}(x,y)} - 1 \right] dxdy \tag{2.18}$$

Associating distance function $d(\cdot,\cdot)$ to a kernel of the form $\mathcal{K}_\epsilon$

$$\mathcal{K}_\epsilon(x,y) = e^{-d(x,y)^2/\epsilon} \tag{2.19}$$

Thus,

$$d(x,y)^2 = -\epsilon \ln \mathcal{K}_\epsilon(x,y) \tag{2.20}$$

Therefore,

$$\begin{aligned} W^2_{2,\epsilon}(\mu_0,\mu_1) &= \inf_{\pi\in\Pi}\left[\int\int_{M\times M} d(x,y)^2\pi(x,y)dxdy - \epsilon H(\pi)\right] \\ &= \inf_{\pi\in\Pi}\left[\int\int_{M\times M} -\epsilon\ln\mathcal{K}_\epsilon(x,y)\pi(x,y)dxdy - \epsilon H(\pi)\right] \\ &= \inf_{\pi\in\Pi}\left[\int\int_{M\times M} \epsilon(-\ln\mathcal{K}_\epsilon(x,y) + \ln\pi(x,y))\pi(x,y)dxdy\right] \\ &= \inf_{\pi\in\Pi}\left[\int\int_{M\times M} \epsilon\left[\ln\frac{\pi(x,y)}{\mathcal{K}_\epsilon(x,y)} - 1\right]\pi(x,y)dxdy + \epsilon\right] \\ &= \epsilon\left[1 + \min_{\pi\in\Pi} KL(\pi|\mathcal{K}_\epsilon)\right] \end{aligned} \tag{2.21}$$

After the algebraic transformation, we notice that computing the entropy-regularized Wasserstein distance is equivalent to computing the smallest KL divergence from a coupling $\pi \in \Pi$ to the kernel $\mathcal{K}_\epsilon$. This gives a new interpretation for regularized transportation problem: the optimal transport problem π is a projection of the distance-based kernel $\mathcal{K}_\epsilon$ onto Π, enforcing marginals while minimizing the loss of information quantified by KL divergence. The minimization is convex; it shed light on a new class of efficient methods to approximate solutions to several generalized optimal transport problem. Jean-David Benamou and Guillaume Carlier and Marco Cuturi and Luca Nenna and Gabriel Peyré (2015) solved problems in the form of $\min_{\gamma\in C} KL(\gamma|\eta)$ using iterative Bregman projection and Dykstra algorithms.

For problems of the form

$$\min_{\gamma\in C} KL(\gamma|\xi) \tag{2.22}$$

where ξ is a given point in $R_+^{N\times N}$ and C is a non-empty intersection of closed convex sets

$$C = \cap_{l=1}^{L} C_l \tag{2.23}$$

- If convex set C_l are affine subspaces, it is possible to solve 2.22 with iterative Bregman Projections.

  Starting from $\gamma^{(0)} = \xi$, one computes

$$\forall n > 0, \gamma^{(n)} = P_{C_n}^{KL}(\gamma^{(n-1)}) \tag{2.24}$$

  (Bregman, 1967) shows that $\gamma^{(n)}$ converges to the unique solution of 2.22

$$\gamma^{(n)} \to P_C^{KL}(\xi) \quad \text{as } n \to \inf \tag{2.25}$$

- In the case where C_l are not affine, Dykstra's algorithm, extended to the KL setting, will converge to the projection.

  Initializing

$$\gamma^{(0)} = \xi \quad \text{and} \quad q^{(0)} = q^{(-1)} = \cdots = q^{(-L+1)} = 1 \tag{2.26}$$

  The series $\gamma^{(n)} \to P_C^{KL}(\xi)$ as $n \to \inf$(Bauschke and Lewis, 2000).

$$\gamma^{(0)} = P_{C_n}^{KL}(\gamma_{n-1} \odot q_{n-L}), \quad \text{and} \quad q^{(n)} = q^{(n-L)} \odot \frac{\gamma^{(n-1)}}{\gamma^{(n)}} \tag{2.27}$$

Using those two algorithms, Benamou et al. investigated in detail the application of the two algorithms described above to the barycenter problem (Jean-David Benamou and Guillaume Carlier and Marco Cuturi and Luca Nenna and Gabriel Peyré, 2015).

Discrete regularized transport $W_\epsilon(p, q) = \min_{\pi \in \Pi(p,q)} < C, \pi > -\epsilon H(\pi)$ can be rewritten as a projection

$$W_\epsilon(\mu_0, \mu_1) = \epsilon \min_{\pi \in \Pi(\mu_0, \mu_1)} KL(\pi|\xi) \quad \text{where} \quad \xi = e^{-\frac{C}{\epsilon}} \tag{2.28}$$

The constraint $\gamma \in \Pi(p, q)$ can be rewrite as two affine subsets of $R_+^{N \times N}$:

$$C_1 = \{\gamma \in R_+^{N \times N}; \gamma 1 = p\} \quad \text{and} \quad C_2 = \{\gamma \in R_+^{N \times N}; \} \tag{2.29}$$

Applying Bregman projection on discrete regularized transport problem

$$P_{C_1}^{KL}(\bar{\gamma}) = diag(\frac{p}{\bar{\gamma} 1})\bar{\gamma} \quad \texttt{and} \quad P_{C_2}^{KL}(\bar{\gamma}) = \bar{\gamma} diag(\frac{q}{\bar{\gamma}^T 1}) \tag{2.30}$$

Writing the Bregman projection as a fix point iteration algorithm, it is exactly the same algorithm as the Sinkhorn algorithm (Cuturi, 2013).

$$\gamma^{(n)} = diag(u^{(n)}) \xi diag(v^{(n)}) \tag{2.31}$$

with

$$u^{(n)} = \frac{p}{\xi v^{(n)}} \quad \texttt{and} \quad v^{(n+1)} = \frac{q}{\xi^T u^{(n)}} \tag{2.32}$$

For barycenter problem

$$\inf_{\mu} \left\{ \sum_{i=1}^{p} \lambda_i W_\epsilon(\nu_i, \mu) \right\} \tag{2.33}$$

it can be rewrite as the KL projection

$$\min \left\{ KL_\lambda(\gamma|\xi) = \sum_{i=1}^{p} \lambda_i KL(\gamma_i|\xi_i); \quad \gamma \in C_1 \cap C_2 \right\} \\ \texttt{where } \forall k, \xi_k = \xi = e^{-C/\epsilon} \tag{2.34}$$

with constraint sets defined by

$$\begin{aligned} C_1 &= \{\gamma = (\gamma_k)_k \in (\Sigma_N)^K; \forall k, \gamma_k^T 1 = \nu_i\} \\ \text{and } C_2 &= \{\gamma = (\gamma_k)_k \in (\Sigma_N)^K; \exists \mu \in R^N, \forall k, \gamma_k 1 = \mu\} \end{aligned} \tag{2.35}$$

The projection $\bar{\gamma} = (\gamma_k)_{k=1}^K = P_{C_2}^{KL_\lambda}(\bar{\gamma})$ satisfies

$$\forall k, \quad \gamma_k = \text{diag}\Big(\frac{p}{\bar{\gamma}_k 1}\Big)\bar{\gamma}_k \qquad \texttt{where } \; p = \Pi_{r=1}^K (\bar{\gamma}_r 1)^{\lambda_r} \tag{2.36}$$

From this, we derive parallel fix point iteration $\gamma^{(n)} = (\gamma_k^{(n)})_k$ which satisfy, for each k

$$\gamma_k^{(n)} = diag(u_k^{(n)}) \xi diag(v_k^{(n)}) \tag{2.37}$$

computing with the iterations

$$u_k^{(n)} = \frac{p^{(n)}}{\eta v_k^{(n)}} \quad \texttt{and} \quad v_k^{(n+1)} = \frac{p_k}{\eta^T u_k^{(n)}} \tag{2.38}$$

where $p^{(n)}$ is the current estimate of the barycenter, computed as

$$p^{(n)} = \Pi_{i=1}^{p} \left(u_k^{(n)} \odot (\xi v_k^{(n)}) \right)^{\lambda_k} \tag{2.39}$$

2.3.3 Convolutional Wasserstein Distance

If one were to choose the heat kernel $\mathcal{H}_t(x, y)$ to associate with $d(\cdot, \cdot)$, and compute through iterative kernel convolutions, one would get a new approximation of the Wasserstein distance called convolutional Wasserstein distance (Solomon et al., 2015).

The heat kernel $\mathcal{H}_t(x, y)$ determines diffusion between $x, y \in M$ after time t. It solves the heat equation $\partial_t f_t = \triangle f_t$ with initial condition f_0 through the map

$$f_t(x) = \int_M f_0(y)\mathcal{H}_t(x, y)dy \tag{2.40}$$

Based on Varadhan's formulation, the distance $d(x, y)$ can be recovered by transferring heat from x to y over a short time interval:

$$d(x, y)^2 = lim_{t\to 0}[-2t \ln \mathcal{H}_t(x, y)] \tag{2.41}$$

Setting $t = \epsilon/2$, we can approximate the kernel K_ϵ in 2.19 as

$$\mathcal{K}_\gamma(x, y) \sim \mathcal{H}_{\epsilon/2}(x, y) \tag{2.42}$$

This indicates $W_{2,H^2_{\epsilon/2}}$ is a diffusion-based approximation of $W^2_{2,\epsilon}$

$$W^2_{2,H_{\epsilon/2}}(\mu_0, \mu_1) = \epsilon\Big[1 + \min_{\pi\in\Pi} KL(\pi|\mathcal{H}_{\epsilon/2})\Big] \tag{2.43}$$

Solving with Bregman projection, proposition 2 and 3 gives us a way to calculate the projection onto constraint sets (Solomon et al., 2015).

2.4 Free Support Barycenter

By observing the formulation of barycenter 2.1, one can see that computing barycenter requires computing a series of Wasserstein distance between different distributions.

It is natural to formulate Wasserstein distance in the form of a linear program: Given that $\mu = \sum_{i=1}^n a_i\delta_{x_i}$, $\nu = \sum_{i=1}^m b_i\delta_{y_i}$ and $M_{XY} = [D(x_i, y_j)^p]_{ij} \in R^{n\times m}$, the Wasserstein distance between μ and ν can be expressed as the optimum of a parametric linear program p defined below

$$p(a, b, M_{XY}) = \min_{T\in U(a,b)} < T, M_{XY} > \tag{2.44}$$

Suppose the barycenter is a probability measure with weights $\{a\}_{i=1}^{n}$ at support locations $\{x_i\}_{i=1}^{n}$. The barycenter formulation can be rewritten as

$$f(a, X) = \sum_{i=1}^{N} \lambda_i * p(a, b_i, M_{XY_i}) \tag{2.45}$$

A standard optimization approach to solve such a non-convex linear equation is to fix one parameter and do minimization on the other one alternatively (Cuturi and Doucet, 2014).

Fixing supports, finding weights for 2.45 becomes the problem $\min_a f(a, X)$. One of the main results from sensitivity analysis of linear program tells us any optimum dual vector α^* of the dual form is a subgradient of $p(a, b, M)$ with respect to α.

The Kantorovich problem is a constrained convex minimization problem, so there exists a natural dual problem for it. The Kantorovich problem admits the dual:

$$d(a, b, M) = \max_{(\alpha,\beta)\in C_M} \alpha^T a + \beta^T b \quad \texttt{where} \;\; C_M = \{(\alpha, \beta) \in R^{n+m} | \alpha_i + \beta_j \leq m_{ij}\} \tag{2.46}$$

Following sensitivity analysis in LP's (Bertsimas and Tsitsiklis, 1997, Eq.7.10), when dual optimal solution coincides with primal optimal solution $d(a, b, M) = p(a, b, M)$. the map $a \to p(a, b, M)$ is a polyhedral convex function, so any optimal dual vector α^* of $d(a, b, M)$ is a subgradient of $p(a, b, M)$ with respect to a. Following the projected subgradient $\alpha = \Sigma_{i=1}^{N} \lambda_i \alpha_i^*$, we minimize f with respect to a.

Now, optimizing over the support locations when fixing weights (i.e., $\min_X f(a, X)$) need some more work:

Let pairwise squared-Euclidean distances between points in these sets $x = diag(X^T X)$ and $y = diag(Y^T Y)$. Then we can use a compact matrix way to compute M_{XY}:

$$M_{XY} = x1_m^T + 1_n y^T - 2X^T Y \in R^{n\times m} \tag{2.47}$$

Then, we can rewrite the Kantorovich problem in compact matrix form:

$$\begin{aligned} < T, M_{XY} > &= < T, x1_m^T + 1_n y^T - 2X^T Y > \\ &= trT^T x1_d^T + trT^T 1_d y^T - 2 < T, X^T Y > \\ &= x^T a + y^T b - 2 < T, X^T Y > \end{aligned} \tag{2.48}$$

Then the barycenter problem can be formulated as:

$$
\begin{aligned}
&\min_X \frac{1}{N}\sum_{i=1}^{N} \min_{T\in U(a,b)} < T, M_{XY} > \\
&= \min_X \frac{1}{N}\sum_{i=1}^{N} \min_{T\in U(a,b)} x^T a + y^T b - 2 < T, X^T Y > \\
&\text{disregarding constant terms in } y \text{ and } b \\
&= \min_X \frac{1}{N}\sum_{i=1}^{N} \min_{T\in U(a,b)} x^T a - 2 < T, X^T Y > \\
&\text{Suppose } T^* \text{ is the optimal solution for } p(a, b, M_{XY}) \\
&= \frac{1}{N}\sum_{i=1}^{N} \min_{T\in U(a,b)} |X diag(a^{1/2}) - YT^{*T} diag(a^{-1/2})|^2 - |YT^{*T} diag(a^{-1/2})|^2
\end{aligned}
\tag{2.49}
$$

Minimizing a local quadratic approximation of p at X yield the Newton update $X \leftarrow YT^T diag(a^{-1})$. Alternatively, minimize locations X and weights a until convergence, we obtain an approximate minimizer of $f(a, X)$ and conclude the algorithm Cuturi and Doucet. One could also add an entropy regularizer to both primal and dual to speed up computation using the Sinkhorn algorithm.

Cuturi and Doucet are not the only ones who took the alternating optimization approach to tackle the problem. Claici and his fellows also adopt a similar method. For fixed weights, applying a single point iteration algorithm akin to Lloyd's algorithm; For fixed-point positions, they estimate gradients and optimize weights using stochastic gradient descent (Claici et al., 2018).

Chapter 3

Our Approach

We saw in Chapter 2, the Wasserstein Barycenter provides us with a natural geometry for probability measures supported on a geometric space. In the language alignment problem, we see the barycenter of all language distributions as a universal language. In Chapter 1, we pointed out the problems of existing methods. In this chapter, we are going to proposing a new framework that improves the quality of translations by guarantee the coherence across all mappings.

We take the probabilistic approach, treating each language $\mathcal{L}_i$ as a probability distribution over its word embeddings:

$$\pi_i = \sum_{j=1}^{n_i} p_j \delta_{x_j^i} \tag{3.1}$$

where p_j is the probability of occurrence of the j-th word x_j^i in language $\mathcal{L}_i$ (often approximated by the relative frequency of word x_j^i in its training documents).

One way to guarantee coherence in mapping is to learn a joint mapping $T : L_i, \cdots, L_m \to R$. The mapping takes a word in each language and returns the probability that all words align with each other.

One caveat here is that not all languages are in the same language space, so we can either train the language space alignment matrix alongside the mapping matrix, or we can use the framework as a post-procedure for the existing bilingual method. After we have some initialization for the space alignment matrix, we can train a joint distribution π for all transformed word embedding distributions $\pi_1, \cdots, \pi_m$.

Formally, we associate each language L_i with a space alignment matrix Q_i, transforming word vectors $\{w_j^{L_i}\}_{j=1}^{n_j}$ to the common language space. To learn the joint probability map π, we

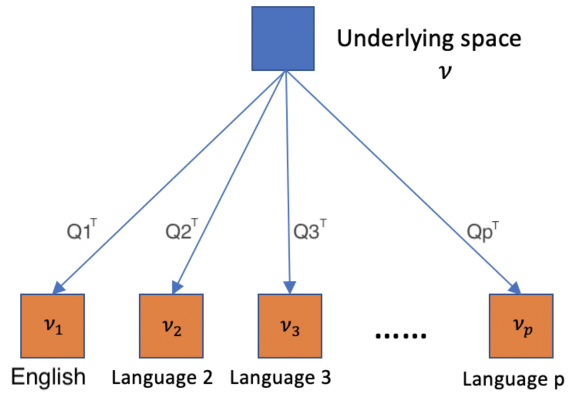

Figure 3.1: Each language will associate with a space alignment matrix projecting word vectors into a common language space. From the potential space, multiplying the word vector with the transpose of word vector matrix will transforming the word vector back to its language space

minimize the objective:

$$\min_{Q_i \in O^d} \min_{\pi \in \Gamma(\pi_1, \cdots, \pi_m)} \int_\pi c(Q_1 x_1, \cdots, Q_m x_m) d\pi(x_1, \cdots, x_m) \tag{3.2}$$

Equation 3.2 targets finding a joint probabilistic map which realizes the smallest cost transporting all pairs of words. The cost function $c(\cdot, \cdots, \cdot)$ is a cost function measuring the similarity between words embedding vectors for words inputted. The more similar, the smaller value it returns. For any cost function $cost(\cdot, \cdot) : L_i \times L_j \to R$, we extend it to multiple inputs in the manner: $c(x_1, \ldots, x_m) = \sum_i \sum_{i \neq j} cost(x_i, x_j)$. With such formulation, words that are close to each other will have higher probability in the joint probability map π.

Solving such a joint map is hard. Fixing the joint map, solving for translation alignment matrices is equivalent to solving 1.25. This gives us hope to optimize 3.2 on joint distribution π. However, there's no current algorithm for learning a joint mapping. Without information on the correlation relationship, such a map is not unique. Not to mention, such a joint probability map may not even exist in some cases.

Although we are not able to compute a joint map for all languages, we can take an alternative method: use the barycenter as an encapsulation of all languages, and imply translations through the barycenter. The concept of a barycenter is closely associated with joint mapping. From the theorem proposed by Agueh and Carlier (2011), we know that the problem of calculating a joint mapping reduce to the barycenter problem. One can quickly realize a barycenter for distributions from the joint distribution for all language distributions.

Such a barycenter should capture all information a joint probability map captures. Through the barycenter, the inferred mapping maintains coherence over the entire space. We interpret the barycenter as a potential "universal" language representing the true meaning of words in all word sets. After computing the barycenter, one can infer translations by calculating the mapping from one language to the barycenter composite with the mapping from barycenter to the other language. We claim that the barycenter is a better choice as the pivot for language alignment.

3.1 Barycenter Approach

When setting the joint cost c as the total sum of all pairwise costs:

$$c(x^1, \ldots, x^m) = \sum_{i,k} \|x^i - x^j\|^2. \tag{3.3}$$

Interestingly, with this choice, we can significantly simplify the numerical computation of the multi-marginal optimal transport.

We reduce the problem into a barycenter problem. Recall the definition of Wasserstein Barycenter ν of m given language distributions $\pi_1, \cdots, \pi_m$. The barycenter ν minimizes the following loss function $J(\nu)$:

$$\inf_{\nu} J(\nu) = \sum_{i=0}^{p} \lambda_i W_2^2(\nu, \pi_i) \tag{3.4}$$

where $\lambda \geq 0$ are the weights, and the (squared) Wasserstein distance W_2^2 is given as:

$$W_2^2(\pi_i, \mu) = \min_{\Pi_i \in \Gamma(\pi_i, \mu)} \int \|x - y\|^2 \mathrm{d}\Pi_i(x, y) \tag{3.5}$$

The notation $\Gamma(\pi_i, \mu)$ denotes all joint probability distribution (i.e. coupling) Π_i with (fixed) marginal distributions π_i and μ. As proven by Agueh and Carlier (2011), with the pairwise distance (3.3), the multi-marginal problem in (3.2) and the barycenter problem in (3.4) are formally equivalent. Hence, from now on we will focus on the latter since efficient computational algorithms for it exist. We use the push-forward notation $(Q_i)_{\#}\pi_i$ to denote the distribution of $Q_i x^i$ when x^i follows the distribution π_i. Thus, we can write our approach succinctly as:

$$\min_{\mu} \min_{Q_i \in \mathcal{O}_d} \sum_{i=1}^{m} \lambda_i \cdot W_2^2[(Q_i)_{\#}\pi_i, \mu], \tag{3.6}$$

where the barycenter μ serves as the pivot language in some common word embedding space. Unlike the arithmetic average in (1.32), the Wasserstein barycenter can have a much larger support (dictionary size) than the m given language distributions.

We can again apply the alternating minimization strategy to solving (3.6): fixing all orthogonal matrices Q_i, we find the Wasserstein barycenter using an existing algorithm of (Cuturi and Doucet, 2014) or (Claici et al., 2018); fixing the Wasserstein barycenter μ, we solve each orthogonal matrix Q_i separately:

$$\min_{Q_i \in \mathcal{O}_d} \min_{\Pi_i \in \Gamma(\pi_i, \mu)} \int \|Q_i x - y\|^2 \mathrm{d}\Pi_i(x, y) \tag{3.7}$$

For fixed coupling $\Pi_i \in \mathcal{R}^{n \times s}$, where s is the dictionary size for the barycenter μ, the integral can be simplified as:

$$\sum_{jl} (\Pi_i)_{jl} \|Q_i x_j^i - y_l\|^2 \equiv -\langle X_i^T \Pi_i Y, Q_i \rangle \tag{3.8}$$

Thus, using the well-known theorem of Schönemann (1966), Q_i is given by the closed-form solution $U_i V_i^T$, where $U_i \Sigma_i V_i^T = X_i^T \Pi_i Y$ is the singular value decomposition.

Compare to using English as the pivot, transiting through the barycenter guarantee coherence over the entire language space. Moreover, it is known that dealing with multiple languages simultaneously has been shown to improve performance on some bilingual tasks by using knowledge learned from other languages. Therefore, this motivates us to devise algorithm 1:

In the following discussion, we use the following notation: L_i denote the i^{th} language inputted, and it contains two part: 1) word embedding matrix C_i with rows as word embedding vectors for words in the vocabulary V_i; and 2) word frequency vector p_i, each entry denote the frequency the corresponding entry in C_i appear in its language. The barycenter will be denoted as C with frequency weights p, and the Optimal transport mapping from the barycenter (C, p) to the i^{th} language (C_i, p_i) will be denoted as T_i.

The current approach (Taitelbaum et al., 2019b) associates an orthogonal matrix Q_i with each language to align all languages to a potential language space. This might be a different space than any language space. In our implementation, we are choosing the language space of the first language L_1 to be the potential language space to align all languages with. Note that choosing a language space to be the potential space instead of learning a potential space to map all languages into does not hurt performance. Since in the setting 1.29, there is an orthogonal constraint on the space mapping matrices. Therefore, all distances and angles for word pairs are preserved, and we can choose any language space to map languages into.

Algorithm 1: barycenter alignment

Input: Language distribution $L_i = (X_i, p_i)_{i=1}^m, p$
Output: Translation for L_k and L_m
for $i = 1;\ i < m;\ i = i + 1$ **do**
    // Preprocess word embeddings ;
    $X_i \leftarrow X_i - X_i.mean(axis = 0)$;
    // Compute intra-distance matrices ;
    $C_i = cdist(X_i)$;
    // Use gromov wasserstein to learn mapping for a small dictionary ;
    $P_i \leftarrow GW(C_1, C_i, p_1, p_i)$;
    $U\Sigma V^T \leftarrow SVD((P_i X_i)^T X_1)$;
    $Q_i \leftarrow UV^T$;
    $X_i' = X_i Q_i$;
while *not converged* **do**
    // Compute the barycenter for realigned languages ;
    $X, p \leftarrow$ barycenter $((X_1', p_1), \cdots, (X_m', p_m))$;
    for $i = 1;\ i < m;\ i = i + 1$ **do**
        // Compute OT mapping from barycenter to each language ;
        $T_i = OT(X, X_i', p, p_i)$;
        // Recompute space alignment matrices ;
        $U\Sigma V^T \leftarrow SVD((T_i X_i)^T X_1)$;
        $Q_i \leftarrow UV^T$;
        // Realign word embedding matrices;
        $X_i' \leftarrow X_i Q_i$;
Return $T_k^T * T_m$

In algorithm 1, we take m language distributions as input $\{L_i = (C_i, p_i)\}_{i=1}^m$. For word frequency p_i, we can either use the Zipf law to predict word frequency (Xavier, 1999) or use the uniform distribution for simplicity.

The main part of the algorithm 1 does iterative optimization on barycenter and language alignment matrix $\{Q_i\}_{i=1}^m$. To train barycenter, we alternatively optimizing on the distribution support locations and the weights on each support point. For the support location, we used a variant of the Lloyd's algorithm; Calculating the optimal weights on the support locations is too costly, so we used the Bregman iteration to compute an approximation of the weights. During Bregman iteration, we save the mapping from the barycenter to each of the language word set

$\{T_i\}_{i=1}^m$. Since the language distributions are discrete distribution, the optimal transport plan between the barycenter C and language matrix C_i is a coupling matrix with one marginal as the weights for C, and the other marginal as weights for C_i. The OT plan from the language C_i to barycenter C is T_i^T. To optimize on the space alignment matrix, we can use Procrustes matching to update all language alignment matrix $\{Q_i\}_{i=1}^m$, with existing language mapping $T_1^T T_i$ for language L_1 and L_i.

To fasten the computation, we took a similar approach as (Alaux et al., 2019), initialize with the Gromov-Wasserstein approach applied to the first 5k vectors and a regularization parameter ϵ of $5e^{-5}$. We choose the first language L_1 to be the language that all other languages align with. After normalizing all language vectors with standard methods in (Dinu and Baroni, 2015), we map all languages into the same language space by multiplying all language embedding vectors with the space alignment matrix.

Algorithm 1 can be treated as a post-procedure for current methods for multilingual alignment. From our experiment, solely use the language alignment matrix already exceeds the benchmark result for state-of-the-art algorithms. However, if one were to use the algorithm to train from scratch, the algorithm takes a while to converge.

3.2 Gromov-Wasserstein Barycenter

Inspired by the Gromov-Wasserstein alignment method (Alvarez-Melis and Jaakkola, 2018), we are motivated to use the Gromov-Wasserstein (GW) barycenter as a pivot.

Recall that in the definition of the Gromov-Wasserstein mapping, we compare distances between words (intra-language distances) in one language to those intra-language distances in another language. To extend the idea with our barycenter framework, we can use the Gromov-Wasserstein barycenter as a pivot for comparing word distances between languages.

The Gromov-Wasserstein barycenter is well defined. Alongside with Gromov-Wasserstein mapping, Peyré et al. (2016) proposed the notion of Gromov-Wasserstein (GW). It's a natural extension of the GW distance:

$$\min_{C \in R^{N \times N}, p \in \Sigma_N} \sum_s \lambda_s GW_\epsilon(C, C_s, p, p_s) \tag{3.9}$$

where C_s is the intra-distance similarity matrix for each language.

The GW barycenter is a Fréchet mean for distances between all words in all languages. To interpret the GW barycenter, we can see it as the desired distance between the meaning of all words in any language.

Algorithm 2: GW-barycenter alignment

Input: Language distribution $L_i = (C_i, p_i)_{i=1}^m, p$
Output: Translation for L_k and L_m
for $i = 1;\ i < m;\ i = i + 1$ **do**
  // Preprocess the word embeddings by normalizing and whitening ;
  $C_i \leftarrow C_i - mean(C_i, axis = 0)$;
  // Calculate the intra-distance matrice ;
  $C_i \leftarrow \cos(C_i, C_i)$
// Calculate the Gromov-Wasserstein barycenter for distributions $(C_s, p_s)_s, p$;
$C \leftarrow$ GW_barycenter$(C_1, p_1, \cdots, C_m, p_m)$;
for $i = 1;\ i < m;\ i = i + 1$ **do**
  // Compute Gromov Wassertein alignment from barycenter to each language ;
  $T_i =$ GW_OT(C, C_i', p, p_i) ;
Return $T_k^T * T_m$

Notice that the problem is a non-convex optimization problem since the GW formulation is non-convex by nature. One way to solve equation 3.9 is to solve the block coordinate relaxation (Peyré et al., 2016). Reintroducing couplings into the reformulation, we can derive an equivalent form for GW-barycenter as 3.9:

$$\min_{C,(T_s)_s} \sum_s \lambda_s(\mathcal{E}_{C,C_S}(T_s) - \epsilon H(T_s)) \tag{3.10}$$

where $\mathcal{E}_{C,C_S}(T_s)$ defined in 1.19.

With the reformulation above, one can solve the problem using block coordinate relaxation. The iterative two-step minimization leads us to a stationary point for the Gromov-Wasserstein barycenter. Optimizing concerning $(T_s)_s$ is fairly straight forward since one can decouple the formulation as S independent GW_ϵ optimizations:

$$\forall s, \quad \min_{T_s \in C_{p,p_s}} \mathcal{E}_{C,C_s}(T_s) - \epsilon H(T_s) \tag{3.11}$$

After computing such $(T_s)_s$, plugging in the values back into reformulation 1.19, and the minimization with respect to C is

$$\min_C \sum_s \lambda_s < \mathcal{L}(C, C_s) \otimes T, T > \tag{3.12}$$

Using 1.3.1 in (Peyré et al., 2016) has shown that if the loss $\mathcal{L}$ can be expressed as and f_1'/h_1' is invertible, then the solution to 3.12 reads

$$C = \left(\frac{f_1'}{h_1'}\right)^{-1} \left(\frac{\sum_s \lambda_s T_s^T h_2(C_s) T_s}{pp^T}\right) \tag{3.13}$$

The intuition behind 3.13 is that Each $T_s^T h_2(C_s) T_s$ is a realigned matrix where T_s acts as a fuzzy permutation (optimal transportation coupling) of both rows and columns of the distance matrix C_s. Then averaging those realigned metrics depending on a different definition of the loss L.

The advantage of training the GW-barycenter as a pivot is that the algorithm converges quickly and requires little to none parameter tuning. For m languages, the GW-barycenter converges within 20 iterations. Therefore, we only need to compute $O(20m)$ GW mappings during training, which is far more efficient than training a mapping for all pairs of languages $O(m^2)$. However, since the computation of GW-barycenter only brings us to a stationary point, there is no guarantee that we can reach the global minimum. Hence its accuracy is not as good as Algorithm 1.

3.3 Hierarchical Approach

When training one single Gromov-Wasserstein barycenter for all languages may hurt performance for some language pairs because of the far distance of origins between the languages. Italian, Spanish, Portuguese, and French are all originated from Latin, whereas German and English are Germanic languages. One way to optimize performance on bilingual translation tasks is to introduce some prior knowledge about languages. The origin of languages is a well-studied field in linguistic.

Figure 3.2 shows a historical view on the development of the language groups in the Indo-European family. Training a barycenter captures shared information across languages so that we can construct a language tree base on language origins. For each node that's not a leaf, we train on that node to be a barycenter for all its children. If we train such a language tree from the bottom up and store mapping during training, mapping for language pairs can be implied by traversing through the tree structure and multiplying all the mappings.

With information on the origin of languages, we can construct the hierarchical barycentric tree 3.3. With the hierarchical tree, we propose a framework for training barycenter. Visiting each node in a DFS manner, we train the nodes from a bottom-up manner. If the node represents

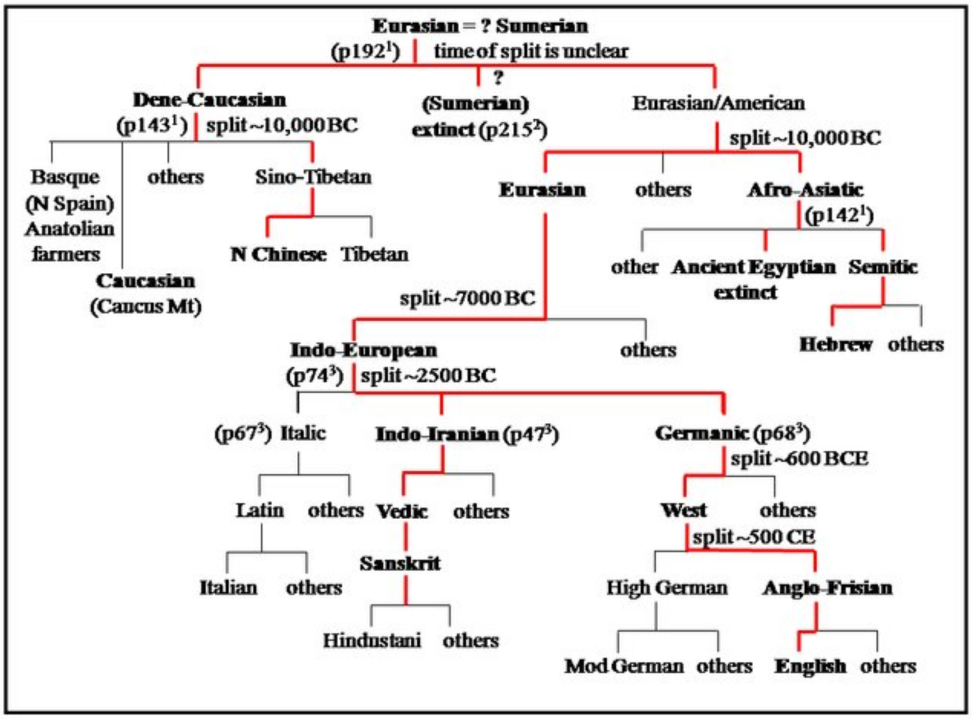

Figure 3.2: Development of language family according to historic information

a language, we skip the training process. If the node is a barycenter (denoted with number in 3.3), then we use either algorithm 1 or algorithm 2 to compute each of barycenter node. At the interpreting step, one can infer language mappings by multiplying the mapping matrices through the edges.

Notice that we can decide the number of support locations for barycenter. In theory, the number of support points for the optimal barycenter could be up to the sum of support points for all distributions. In the experiment chapter 4, we are investigated on the number of support points and found out that choosing the number of support points to be 2 times the average number of support points of all input distributions is the best choice evaluating on the efficiency and accuracy.

Notice that use Algorithm 1 fall into the framework since it's just a single level language tree with all languages under one common root node.

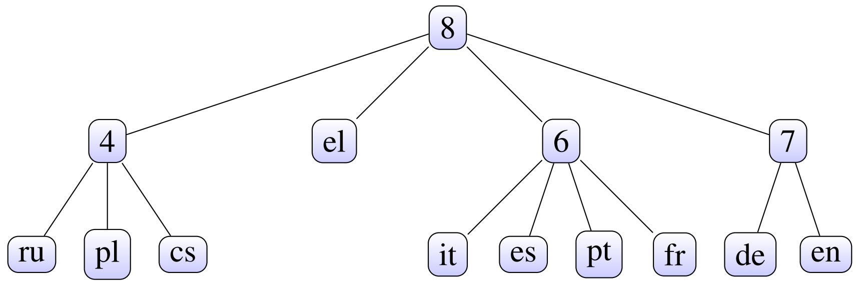

Figure 3.3: Language Tree in a tree structure

Algorithm 3: GW-hierarchical barycenter alignment

Input: Language distribution $(L_s, p_s)_s, p, root$
Output: Translation for all pairs of languages

for *each language* L_i **do**
  Preprocess the word embeddings by normalizing and whitening ;
  Calculate the intra-distance matrice C_s for L_s ;

Initialize a stack S ;
Perform a BFS on the lingual tree inputted $root$ and put all nodes into S;
while S *is not empty* **do**
  pop node q from stack S ;
  if q *is not a language node* **then**
    train the Gromov-Wasserstein barycenter for distributions of $q's$ child nodes $(C_{s'}, p_{s'})_{s'}, p$;
    Save the barycenter trained ;
    During the training process, save the mapping from the barycenter to all languages $(T_s)_s$;

Imply translations with the mapping $T_i^T * T_j$ using CSLS

Chapter 4

Experiments

In chapter 3, we proposed two different methods to align multiple languages. In chapter 2, we have discussed the properties of barycenters and methods to compute efficiently. In this section, we are going to compare their evaluation against previous methods in the literature.

Currently, there are two standard benchmarks for cross-lingual word embedding alignment tasks: MUSE (Conneau et al., 2017) and Dinu (Dinu and Baroni, 2015). Before 2014, to evaluate the word alignment method, one has to create some set of evaluation dictionary for translations. Usually, those dictionaries are created using Google translate results or phrase tables of machine translation systems. The problem with such dictionaries is that they do not take account of polysemy of words, and therefore, leading to wrong evaluation of the quality of word embeddings.

In 2018, Conneau et al. created and released their high-quality dictionary of up to 100k pairs of words as part of MUSE library, (Conneau et al., 2017). The MUSE library is a high-quality dictionary containing up to 100k pairs of words. It now becomes a standard benchmark for language alignment problems. The Dinu dataset is another benchmark for evaluation, is a relatively small dataset with only four language pairs (English to Finnish, English to German, English to Spanish, and English to Italian). The Dinu dataset is not suitable to evaluate our method, because the emphasis of this project shows an improvement in cross-lingual evaluation, and the dataset does not have enough languages. Therefore, we conducted experiment on 6 European languages: English, French, Spanish, Italian, Portuguese, and German, and evaluate our algorithm against the MUSE benchmark (Conneau et al., 2017). We choose to conduct experiment on these 6 European languages because MUSE dataset contains a direct translation for any pair of languages in this set.

We also conducted an experiment with the XLING dataset (Glavaš et al., 2019) with a more diverse set of languages: Croatian (HR), English (EN), Finnish (FI), French (FR), German (DE), Italian (IT), Russian (RU), and Turkish (TR). In this set of languages, we have languages coming from three different Indo-European branches, as well as two non-Indo-European languages (FI from Uralic and TR from Turkic family). The results are captured in Table 4.8.

Baselines We are going to compare our results to the three baselines:

1. Using English as a pivot to translate languages
2. state-of-the-art bilingual alignment method: Gromov-Wasserstein alignment (Alvarez-Melis and Jaakkola, 2018)
3. state-of-the-art multilingual alignment method (Alaux et al., 2019)

The hardness to beat for the baselines is the same as the sequence listed above. Transiting through English suffers from the loss of information when the language pair we want to translate does not involve English. The Gromov-Wasserstein bilingual alignment is the state-of-the-art method for bilingual alignment, so it is harder to beat baseline. The state-of-art multilingual alignment method is kept by (Taitelbaum et al., 2019b). It is a wide-know fact that one can improve the performance of language translation using information from all languages.

Implementation Detail As reported in (Conneau et al., 2017), using fastText embeddings trained on Wikipedia obtain a significant boost in performance for word translation tasks. In all our experiments, we use fastText word embeddings to get a continuous vector representation for words in all languages. At preprocess step, we normalize all word embeddings for different languages by centering all word embedding vectors around mean and potentially, scaling them to have length 1. After using algorithm 1 or 2, we predict translations by finding the top k values in the mapping matrix P for the row with the index of the source word. To speed up the computation, we took a similar approach as Alaux et al. (2019) and initialized with the Gromov-Wasserstein approach applied to the first 5k vectors and a regularization parameter ϵ of $5e^{-5}$; whereas Alaux et al. (2019) used the first 2k vectors. The initial locations of barycenter support points are sampled randomly from the standard normal distribution and the number of support points is specified by the user. After normalizing all language vectors with standard methods in Dinu and Baroni (2015), we map all languages into the same language space by multiplying all language embedding vectors with the space alignment matrix. Evaluating results against the MUSE dataset, we measure how many times one of the correct translations of a source word is retrieved, and report precision@k for $k = 1, 10$. The precision@k accounts for a fraction of pairs for which the correct translation of the source words is in the k-th nearest neighbors.

Gromov-Wasserstein Barycenter Analysis In this set of experiment, we are comparing the accuracy for all pairs of languages for the cases when 1) we English as a pivot to translate languages; 2) with Gromov-Wasserstein of all languages as a pivot to infer language translation; and 3) taking the hierarchical approach to compute Gromov-Wasserstein barycenter for language trees. This set of evaluation runs We show results in Table 4. For each bilingual pair, the best performance is highlighted with bold.

As we can see, the bilingual mapping achieved by transiting through Gromov-Wasserstein barycenter is similar to transiting through English as a pivot. The advantage of the algorithm 2 is it is computationally cheaper. The notion of having a probability coupling indicating the mapping between languages is convenient in the sense that we can multiply mappings to infer translation when we have the mapping of languages with the same language L. In other words, the mapping for language L_i and language L_j is $T_i^T T_j$ where T_i is the mapping from L to L_i and mapping T_j is mapping from L to L_j.

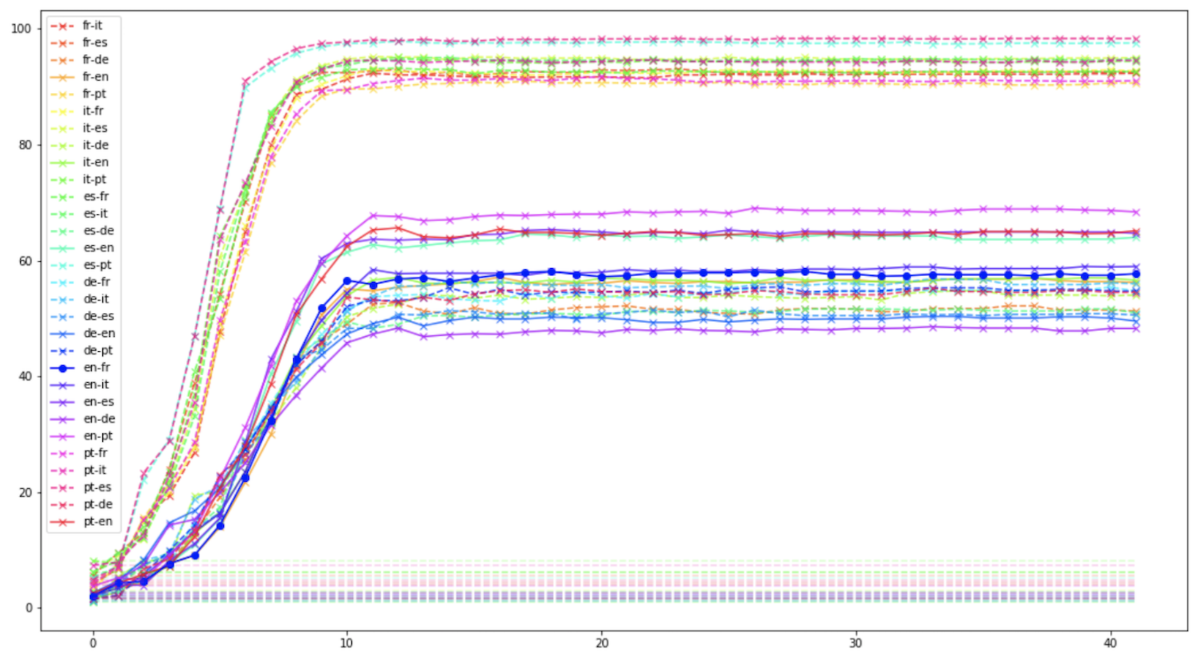

Figure 4.1: The P@1 accuracy for each of language pair during GW-barycenter training. The x-axis denotes the number of iteration the alternate minimization on 1.19

In Figure 4.1, we can see that the accuracy converges around the 10^{th} iteration, and further training does not improve the results anymore. This is true for any random initialization of the GW-barycenter.

The resulting performance is not as good as expected, partially because the GW-barycenter

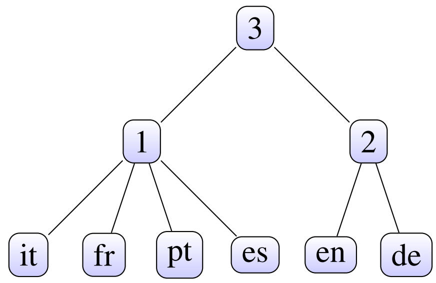

Figure 4.2: The language tree of European languages. Nodes under the same tree node are from less distant origins than those are not. This tree is a subtree trimmed from 3.3

algorithm may not converge to the global optimal. Another reason is that not all languages have the same set of words with the same meanings. Next, we are going to show the performance of the barycenter approach described in 1 as it does not suffer from these problems.

Ablation Study There are a few parameters to choose during barycenter computation. One of the parameters is the number of support locations. In theory, the optimal barycenter distribution could have support locations with the number up to the sum of the total number of support locations of all input distributions.

In table 4, we show the impact on translation performance when we have a different number of support locations. Let n_j be the number of words we have in language L_j. We picked three most representative cases: the average number of words avg $= \sum_{j=1}^{m} n_j/m$, 2 times the average number of words 2avg $= 2\sum_{j=1}^{m} n_j/m$, and the total number $m$avg $= \sum_{j=1}^{m} n_j$.

As we increase the number of support locations for barycenter distribution, we can see from table 4 the performance for language translation improves. However, when we increase the number of support for barycenter, the algorithm becomes more costly. Therefore, evaluating the tradeoff between accuracy and computation power, we decided to choose the support location to be 10000 (2 times the average number of words in all languages).

Next we compare the results of 1) barycenter approach with 2avg number of supports to the 2) baseline # 2 - Gromov Wasserstein alignment on two languages directly, 3) the case where we do not optimize on support location points (Lloyd's algorithm), and 4) hierarchical barycenter on the language origin tree 4. With those different setups, we compare the accuracy to compare The results are contained in table 4.4. The best result we achieved is to use Algorithm 1 with one single barycenter tree node. The result tells us that using the information of all languages during training any language pairs does help improve translation mapping.

We also conducted a set of experiments to determine whether the inclusion of distant languages increases bilingual translation accuracy. Excluding two non-Indo-European languages

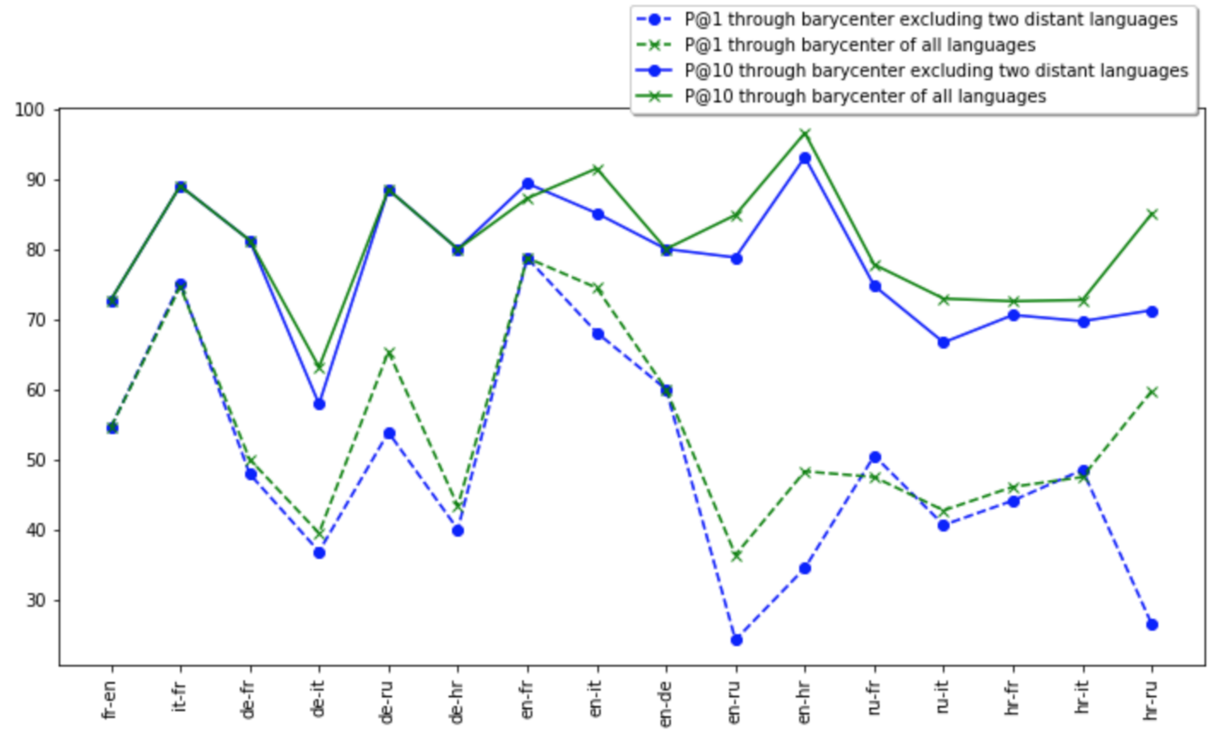

Figure 4.3: This graph shows the accuracy of bilingual translation pairs. The green lines indicate translation accuracy going through the barycenter of all languages (HR, EN, FI, FR, DE, RU, IT, TR), and the blue one is by going through the barycenter of (HR, EN, FR, DE, IT, RU).

Finnish and Turkish, we calculated the barycenter of Croatian (HR), English (EN), French (FR), German (DE), and Italian (IT). Graph 4.3 contains results for common bilingual pairs. Green lines show the bilingual translation accuracy when translating through the barycenter for all languages including Finnish and Turkish, whereas blue lines indicate the accuracy of translations that use the barycenter of the five Indo-European languages.

Now we are going to compare the results with the third baseline: multilingual framework by Alaux et al. We insert our results for Algorithm 1 into their results table 4.7. Looking at results for all pairs of languages side by side, we can see that the performance of our algorithm exceeds theirs in all cases except for pairs of languages involving English. Even when only comparing the pair of languages containing English, our barycenter approach has shown significant improvement. We compare all pairs of languages from or to English to all current methods, both unsupervised and supervised. Putting all results into one single table 4.6, we can see that our

German Word	English Translation	GW Prediction	BA Prediction
münchen	munich	london; dublin; oxford; birmingham; wellington glasgow; edinburgh; cambridge; toronto; hamilton	cambridge; oxford; **munich**; london; birmingham bristol; edinburgh; dublin; hampshire; baltimore
sollte	should	would; could; might; will; needs; supposed; put; willing; wanted; meant;	would; could; will; supposed; **should** might; meant; needs; expected; able
lassen	let	make; to; able; allow; find seek; prove; encourage; identify; try	**let**; make; able; allow; to continue; choose; give; encourage; decide
rahmen	frame; framework	joint; programs; aimed; panel; exercise sponsored; conducted; program; initiative; launched	programme; program; part; programmes; conducted **framework**; programme; part; joint; funded
dieser	this	another;whose;the; which;a itself; having; that; of; thus	the;of;whose;which; that another; **this**; one; latter;its

Table 4.1: German-to-English translation prediction comparing results by 1) using GW alignment to imply direct bilingual mapping and 2) using Barycenter Alignment method discribed in Algorithm 1.

result exceeds current all unsupervised methods for most cases. On average, we see a 2 percent bump in current methods. In the case of translating English to Russian, our method even exceeds the supervised method.

Examples of Predicted Translation Our barycenter approach infers a ”potential universal language” from input languages. Transiting through that universal language, we infer translation for all pairs of languages. From the experimental results, we can see that our approach is clearly at an advantage. Our algorithm has a clear advantage over all other methods, benefiting from using the information of all languages. Our method is capable of incorporating both the semantic and syntactic information of one word. For example, the top ten of our predicted English translation for the German word *münchen*, are ”cambridge”,”oxford”,”munich”,”london”, ”birmingham”,”bristol”,”edinburgh”,”dublin”,”hampshire”,”baltimore”. In this case, we hit the English translation ”munich”. What’s more important in this example is that all predicted English words are the name of some city. Therefore, our method is capable of implying *münchen* is a city name. Another Example to show is German word *sollte*, which means ”should” in English. The top ten translation predicted are ”would”, ”could”, ”will”, ”supposed”, ”should”, ”might”, ”meant”, ”needs”, ”expected”, ”able”. The top five words we predicted for *sollte* are syntactically correct - ”would”, ”could”, ”will”, ”should”, ”might” are all modal verbs.

In table 4, we picked a couple of German to English results and compared the results by Gromov-Wasserstein direct bilingual alignment, and using Algorithm 1.

Table 4.2: The table compares the translation pair accuracy when we 1) use the barycenter as a pivot, 2) use GW-barycenter for all languages as a pivot to translate bilingual pairs (Algorithm 2), and 3) infer translations through mappings on edges of a hierarchical barycenter tree described in Algorithm 3

	English pivot		barycenter pivot		Hierarchical barycenter pivot	
	P@1	P@10	P@1	P@10	P@1	P@10
pt-it	88.13	97.12	89.61	97.75	**90.52**	**97.86**
pt-es	95.51	99.11	**95.59**	**99.17**	95.54	99.05
pt-fr	88.54	97.19	**90.03**	**97.43**	89.52	97.13
pt-de	71.85	93.27	**73.12**	92.44	72.9	**93.43**
pt-en	**81.14**	**94.17**	80.56	93.86	80.23	93.75
it-pt	87.73	97.35	89.13	**97.7**	**89.44**	97.42
it-es	89.76	97.58	91.28	97.51	**92.05**	**97.85**
it-fr	90.87	97.82	**91.67**	97.89	91.59	**97.93**
it-de	72.53	93.01	**72.85**	92.26	**72.85**	**93.23**
it-en	**80.38**	**93.3**	78.74	92.81	78.74	92.65
es-pt	91.83	**98.32**	92.4	98.18	**92.43**	98.09
es-it	87.93	96.89	88.35	96.57	**88.99**	**97.11**
es-fr	90.48	98.14	**91.37**	**98.21**	91.0	97.84
es-de	74.65	**92.96**	**75.16**	92.45	74.65	92.45
es-en	**81.52**	**94.79**	81.27	94.27	81.47	94.52
fr-pt	85.79	96.63	87.25	**96.85**	**87.35**	96.66
fr-it	87.2	97.5	87.79	97.47	**88.1**	**97.54**
fr-es	88.78	97.9	**90.41**	**98.12**	90.07	97.64
fr-de	73.44	92.37	**73.65**	**92.84**	71.56	91.32
fr-en	**82.2**	**94.19**	80.55	93.96	79.87	93.56
de-pt	72.19	93.71	**74.06**	92.5	72.85	**93.76**
de-it	73.27	93.59	73.6	93.32	**74.48**	**93.81**
de-es	72.14	**91.84**	**72.74**	91.6	71.63	91.8
de-fr	75.08	**93.45**	**76.61**	93.4	72.91	92.19
de-en	**72.85**	**91.06**	71.71	89.81	72.34	90.98
en-pt	82.97	94.57	82.41	**94.92**	**83.06**	94.48
en-it	**80.56**	**94.03**	79.43	93.93	79.53	93.9
en-es	82.23	94.7	81.38	94.58	**82.35**	**94.97**
en-fr	**81.76**	94.15	81.33	**94.37**	81.15	93.87
en-de	**71.67**	**90.55**	69.56	89.87	71.15	**90.55**

Table 4.3: This table contains accuracy for translation pairs with different number of support locations for barycenter. The column names are the number of support locations used in experiment. In our experiment setup, we have 5000 words in each language. The column 5000 is average number of support for all language distributions; 10000 is 2times average; sum is the sum of number of supports for all distributions.

	5000		10000		sum	
	P@1	P@10	P@1	P@10	P@1	P@10
it-es	91.72	97.98	**92.29**	98.01	92.26	**98.25**
it-fr	**92.43**	98.07	**92.43**	98.14	92.32	**98.25**
it-pt	89.86	97.87	**90.0**	97.84	89.58	**97.91**
it-en	81.59	93.34	81.93	93.77	**82.21**	**94.27**
it-de	75.54	93.44	75.86	**93.82**	**75.91**	93.71
es-it	89.22	97.3	**89.47**	97.43	**89.47**	**97.5**
es-fr	**92.08**	98.21	91.63	**98.33**	91.89	**98.33**
es-pt	92.57	98.12	**92.88**	**98.35**	92.48	98.23
es-en	83.48	94.88	83.53	**95.48**	**84.11**	95.37
es-de	78.25	94.23	78.08	**94.74**	**78.55**	94.66
fr-it	**88.52**	**97.71**	88.28	**97.71**	88.31	97.64
fr-es	90.55	97.93	90.74	**98.04**	**90.85**	**98.04**
fr-pt	88.19	96.95	**88.51**	**97.08**	88.09	96.98
fr-en	82.66	94.68	82.63	94.42	**83.14**	**94.85**
fr-de	75.69	93.41	**76.53**	93.41	**76.53**	**93.88**
pt-it	**90.77**	97.93	**90.77**	**97.96**	90.38	**97.96**
pt-es	95.68	99.17	**95.92**	**99.32**	95.74	**99.32**
pt-fr	90.81	97.6	**91.01**	**97.87**	90.57	97.67
pt-en	83.14	94.39	83.0	**94.64**	**83.19**	94.61
pt-de	77.7	94.43	77.32	94.32	**77.98**	**94.65**
en-it	82.0	94.51	81.86	94.58	**82.24**	**94.82**
en-es	84.11	95.16	84.14	95.28	**84.68**	**95.43**
en-fr	83.03	94.4	82.94	94.67	**83.44**	**94.83**
en-pt	84.93	94.88	84.68	**95.29**	**85.34**	95.26
en-de	73.47	90.91	73.99	91.46	**74.63**	**91.83**
de-it	**78.09**	93.92	78.04	94.52	77.66	**94.74**
de-es	75.85	93.31	76.16	**93.83**	**77.4**	93.39
de-fr	78.09	93.88	79.09	93.77	**79.25**	**94.56**
de-pt	77.07	94.2	77.5	94.14	**78.49**	**95.02**
de-en	75.19	91.29	75.74	91.98	**76.19**	**92.41**

Table 4.4: Accuracy results for translation pairs between all pairs of languages from all different methods we proposed in this thesis. The column GW-benchmark contain results from Gromov-Wasserstein direct bilingual alignment. Unweighted is the barycenter approach without optimizing on support location weights. Hierarchical contains results from traversaling through edges and infer translation mapping through hierarchical barycenters. The weighted column is what Algorithm 1 returns, optimizing both on support locations and weights on the support.

	GW benchmark		unweighted		hierarchical		weighted	
	P@1	P@10	P@1	P@10	P@1	P@10	P@1	P@10
it-es	**92.63**	98.05	91.52	97.95	92.49	**98.11**	92.29	98.01
it-fr	91.78	98.11	91.27	97.89	**92.61**	**98.14**	92.43	**98.14**
it-pt	89.47	97.35	88.22	97.25	89.89	**97.87**	**90.0**	97.84
it-en	80.38	93.3	79.23	93.18	79.54	93.21	**81.93**	**93.77**
it-de	74.03	93.66	74.41	92.96	73.06	92.26	**75.86**	**93.82**
es-it	89.35	97.3	88.8	97.05	**89.73**	**97.5**	89.47	97.43
es-fr	**91.78**	98.21	91.34	98.03	91.74	98.29	91.63	**98.33**
es-pt	92.82	98.32	91.83	98.18	92.65	**98.35**	**92.88**	**98.35**
es-en	81.52	94.79	82.43	94.63	81.63	94.27	**83.53**	**95.48**
es-de	75.03	93.98	76.47	93.73	74.86	93.73	**78.08**	**94.74**
fr-it	88.0	97.5	87.55	97.19	**88.35**	97.64	88.28	**97.71**
fr-es	90.3	97.97	90.18	97.68	90.66	**98.04**	**90.74**	**98.04**
fr-pt	87.44	96.89	86.7	96.79	88.35	**97.11**	**88.51**	97.08
fr-en	82.2	94.19	81.26	94.25	80.89	94.13	**82.63**	**94.42**
fr-de	74.18	92.94	74.07	92.73	74.44	92.68	**76.53**	**93.41**
pt-it	90.62	97.61	89.36	97.75	90.59	**98.17**	**90.77**	97.96
pt-es	**96.19**	99.29	95.36	99.08	96.04	99.23	95.92	**99.32**
pt-fr	89.9	97.57	90.1	97.43	90.67	97.74	**91.01**	**97.87**
pt-en	81.14	94.17	81.42	94.14	81.42	93.86	**83.0**	**94.64**
pt-de	74.83	93.76	75.94	93.21	74.45	93.1	**77.32**	**94.32**
en-it	80.84	93.97	79.88	93.93	80.25	93.76	**81.86**	**94.58**
en-es	82.35	94.67	83.05	94.79	81.62	94.82	**84.14**	**95.28**
en-fr	81.67	94.24	81.86	94.33	81.42	93.99	**82.94**	**94.67**
en-pt	83.03	94.45	82.72	94.64	82.25	94.79	**84.68**	**95.29**
en-de	71.73	90.48	72.92	90.76	71.88	90.42	**73.99**	**91.46**
de-it	75.41	94.3	76.4	93.87	75.19	93.65	**78.04**	**94.52**
de-es	72.18	92.64	74.21	92.6	73.58	92.48	**76.16**	**93.83**
de-fr	77.14	93.29	77.93	93.61	77.14	93.51	**79.09**	**93.77**
de-pt	74.38	93.71	74.99	93.54	74.22	93.81	**77.5**	**94.14**
de-en	72.85	91.06	74.36	91.21	72.17	90.81	**75.74**	**91.98**
average	82.84	95.26	82.86	95.15	82.79	95.18	84.23	95.67

Table 4.5: Pairs of languages in multilingual alignment problem results for English, German, French, Spanish, Italian, and Portuguese. All reported results are precision@1 percentage. The method achieving the highest precision for each bilingual pair is highlighted in bold. Methods we are comparing to in the table are: Gromov-Wasserstein alignment (GW) (Alvarez-Melis and Jaakkola, 2018); bilingual alignment with multilingual auxiliary information (MPPA) (Taitelbaum et al., 2019a); Multilingual pseudosupervised refinement method (Chen and Cardie, 2018); multilingual alignment method (UMH) (Alaux et al., 2019).

	it-es	it-fr	it-pt	it-en	it-de	es-it	es-fr	es-pt	es-en	es-de
GW	**92.63**	91.78	89.47	80.38	74.03	89.35	91.78	92.82	81.52	75.03
PA	87.3	87.1	81.0	76.9	67.5	83.5	85.8	87.3	82.9	68.3
MAT+MPPA	87.5	87.7	81.2	77.7	67.1	83.7	85.9	86.8	83.5	66.5
MAT+MPSR	88.2	88.1	82.3	77.4	69.5	84.5	86.9	87.8	83.7	69.0
UMH	87.0	86.7	80.4	79.9	67.5	83.3	85.1	86.3	**85.3**	68.7
BA	92.32	**92.54**	**90.14**	**81.84**	**75.65**	**89.38**	**92.19**	**92.85**	83.5	**78.25**
	fr-it	fr-es	fr-pt	fr-en	fr-de	pt-it	pt-es	pt-fr	pt-en	pt-de
GW	88.0	90.3	87.44	82.2	74.18	90.62	**96.19**	89.9	81.14	74.83
PA	83.2	82.6	78.1	82.4	69.5	81.1	91.5	84.3	80.3	63.7
MAT+MPPA	83.1	83.6	78.7	82.2	69.0	82.6	92.2	84.6	80.2	63.7
MAT+MPSR	83.5	83.9	79.3	81.8	71.2	82.6	92.7	86.3	79.9	65.7
UMH	82.5	82.7	77.5	83.1	69.8	81.1	91.7	83.6	82.1	64.4
BA	**88.38**	**90.77**	**88.22**	**83.23**	**76.63**	**91.08**	96.04	**91.04**	**82.91**	**76.99**
	en-it	en-es	en-fr	en-pt	en-de	de-it	de-es	de-fr	de-pt	de-en
GW	80.84	82.35	81.67	83.03	71.73	75.41	72.18	77.14	74.38	72.85
PA	77.3	81.4	81.1	79.9	73.5	69.5	67.7	73.3	59.1	72.4
MAT+MPPA	78.5	82.2	82.7	81.3	74.5	70.1	68.0	75.2	61.1	72.9
MAT+MPSR	78.8	82.5	82.4	81.5	74.8	72.0	69.6	76.7	63.2	72.9
UMH	78.9	82.5	82.7	82.0	**75.1**	68.7	67.2	73.5	59.0	75.5
BA	**81.45**	**84.26**	**82.94**	**84.65**	74.08	**78.09**	**75.93**	**78.93**	**77.18**	**75.85**

Table 4.6: Accuracy for all current methods, both supervised and unsupervised, for bilingual alignment evaluating on the MUSE benchmark. All approaches use a CSLS criterion. "ref" refers to the refinement method of Conneau et al. (Conneau et al., 2017). The results are directly copied from (Alaux et al., 2019). The best overall accuracy is underlined, and in bold among unsupervised methods.

	en-es		en-fr		en-it		en-de		en-ru		Avg.
	→	←	→	←	→	←	→	←	→	←	
`supervised, bilingual`											
Proc.	80.9	82.9	91.0	82.3	75.3	77.7	74.3	72.4	51.2	64.5	74.3
GeoMM	81.4	85.5	82.1	84.1	-	-	74.7	76.7	51.3	67.6	-
RCSLS	84.1	86.3	83.3	84.1	79.3	81.5	79.1	76.3	57.9	67.2	77.9
`unsupervised, bilingual`											
GW	81.7	80.4	81.3	78.9	78.9	75.2	71.9	72.8	45.1	43.7	71.0
Adv + ref	81.7	83.3	82.3	82.1	77.4	76.1	74.0	72.2	44.0	59.1	73.2
ICP + ref	82.1	84.1	82.3	82.9	77.9	77.5	74.7	73.0	47.5	61.8	74.4
W-Proc + ref	82.8	84.1	82.6	82.9	-	-	75.4	73.3	43.7	59.1	-
UMH bil.	82.5	84.9	82.9	83.3	79.4	79.4	74.8	73.7	45.3	62.8	74.9
`unsupervised, multilingual`											
MAT+MPSR	82.5	83.7	82.4	81.8	78.8	77.4	74.8	72.9	-	-	-
UMH multi.	82.4	85.1	82.7	83.4	78.1	79.3	75.5	74.4	45.8	64.9	75.2
BA	84.68	84.11	83.44	83.14	82.24	82.21	74.63	76.19	56.53	63.76	77.09

Table 4.7: Comparing the accuracy resulting from UMH with our results returned from the barycenter algorithm 1 for all pairs of languages. UMH denote the result of Alaux et al. in (Alaux et al., 2019), and BA is short for Barycenter Alignment

→	en		fr		es		it		pt		de	
	UMH	BA	UMH	BA	UMH	BA	UMH	BA	UMH	BA	UMH	BA
en	-	-	82.7	82.94	82.5	84.14	78.9	81.86	82.0	84.68	75.1	73.99
fr	83.1	82.63	-	-	82.7	90.74	82.5	88.28	77.5	88.51	69.8	76.53
es	85.3	83.53	85.1	91.63	-	-	83.3	89.47	86.3	92.88	68.7	78.08
it	79.9	81.93	86.7	92.43	87.0	92.29	-	-	80.4	90.0	67.5	75.86
pt	82.1	83.0	83.6	91.01	91.7	95.92	81.1	90.77	-	-	64.4	77.32
de	75.5	75.74	73.5	79.09	67.2	76.16	68.7	78.04	59.0	77.5	-	-

Table 4.8: Evaluated on XLING dataset. Accuracy are measured with mean average precision (MAP).

	en-de	it-fr	hr-ru	en-hr	de-fi	tr-fr	ru-it	fi-hr	tr-hr	tr-ru
PROC (1k)	0.458	0.615	0.269	0.225	0.264	0.215	0.360	0.187	0.148	0.168
PROC (5k)	0.544	0.669	0.372	0.336	0.359	0.338	0.474	0.294	0.259	0.290
PROC-B	0.521	0.665	0.348	0.296	0.354	0.305	0.466	0.263	0.210	0.230
RCSLS (1k)	0.501	0.637	0.291	0.267	0.288	0.247	0.383	0.214	0.170	0.191
RCSLS (5k)	0.580	0.682	0.404	0.375	0.395	0.375	0.491	0.321	0.285	0.324
GW	0.667	0.751	**0.683**	0.123	0.454	0.485	0.508	**0.634**	**0.482**	0.295
BA	**0.683**	**0.799**	0.667	**0.646**	**0.508**	**0.513**	**0.512**	0.601	0.481	**0.355**

Chapter 5

Conclusion and Discussion

In this section, I summarize the contributions made in this thesis and discuss possible future directions.

In the thesis, we are solving the unsupervised version of the multilingual alignment problem. In Chapter 1, we gave a literature review on existing bilingual methods and introduced a couple of multilingual framework researchers have devised to extend bilingual alignment to the multilingual case. This is also the chapter where we formally defined the problem and pointed out the shortcoming for some current methods.

Previous works have shown that using information from other languages improve bilingual translations. The problem of the most existing multilingual framework is the lack of transitive enforcement for pivot language. So we proposed a new pivot that guarantees coherence over the language space where language word vectors for all languages are lying in. The algorithm we are proposing trains all languages simultaneously and use intermediate result saved during the training process to infer translations for all pairs of languages. At the core of our algorithm lies the computation of barycenter for all languages. The barycenter we computed for all languages can be interpreted as a potential universal language, capturing information of all languages. We spared a chapter to introduce the concept of barycenter and existing algorithms to compute it in Chapter 2. The algorithm we proposed beats the current state-of-the-art method for bilingual alignment. We discuss the results and evaluation method in Chapter 4.

This work can be extended in several directions in the future. First, we notice that the Gromov-Wasserstein barycenter algorithm relies heavily on its initialization. Current methods only bring us to a stationary point instead of a global minimum. Devising a new convex relaxation will bring us a new algorithm that may converge to a global minimum.

In the barycenter computation, we need to decide the weights we are going to put on each input distribution. The other problem we are interested in investigating in is: whether a hierarchical barycenter is equivalent to the joint barycenter is an interesting question. Given a hierarchical tree, can we assign different weights to different nodes and the root barycenter will be equal to the joint barycenter? Simplify the question, given distributions $(X_i, p_i)_{i=1}^m$ and a weights vector $\{\alpha\}_{i=1}^m$, can we find two different vectors $\{\beta_1\}_{i=1}^{b_1}$, $\{\beta_2\}_{i=1}^{b_2}$, both summing to 1, where $b_1 + b_2 = m$, and $\arg\min_X \sum_i \alpha_i W_\epsilon(X, X_i) = \arg\min_X (1/2 * W_\epsilon(X, \arg\min_A \sum_{i=1}^{b_1} \beta_{1i} W_\epsilon(A, X_i)) + 1/2 * W_\epsilon(X, \arg\min_B \sum_{i=1}^{b2} \beta_{2i} W_\epsilon(B, X_{i+b_1})))$. If such a decomposition exists, a faster barycenter algorithm may be possible.

References

[1] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. "Distributed Representations of Words and Phrases and their Compositionality". In: *Neural Information Processing Systems (NIPS)*. 2013, pp. 3111–3119 (cit. on pp. 1, 2, 13).

[2] Reinhard Rapp. "Identifying Word Translations in Non-parallel Texts". In: *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*. ACL '95. Association for Computational Linguistics, 1995, pp. 320–322 (cit. on pp. 1, 3).

[3] Pascale Fung. "Compiling bilingual lexicon entries from a non-parallel English-Chinese corpus". In: *Third Workshop on Very Large Corpora*. 1995 (cit. on pp. 1, 3).

[4] Yuan Zhang, David Gaddy, Regina Barzilay, and Tommi Jaakkola. "Ten pairs to tag-Multilingual POS tagging via coarse mapping between embeddings". In: *Association for Computational Linguistics*. Association for Computational Linguistics. 2016 (cit. on p. 1).

[5] Georgiana Dinu and Marco Baroni. "Improving zero-shot learning by mitigating the hubness problem". In: *3rd International Conference on Learning Representations (ICLR)*. 2015 (cit. on pp. 1, 3, 8, 36, 41, 42).

[6] Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. "Offline Bilingual Word vectors, Orthogonal Transformations and the Inverted Softmax". In: 2017 (cit. on pp. 1, 13, 15, 16).

[7] Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. "Word Translation Without Parallel Data". In: *6th International Conference on Learning Representations (ICLR)*. Vol. abs/1710.04087. 2017 (cit. on pp. 2, 4, 5, 9, 12, 15, 16, 41, 42, 51).

[8] Hagai Taitelbaum, Gal Chechik, and Jacob Goldberger. "Multilingual Word Translation using Auxiliary Languages". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019 (cit. on pp. 2, 16, 17, 50).

[9] Ndapandula Nakashole and Raphael Flauger. "Knowledge Distillation for Bilingual Dictionary Induction". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Sept. 2017, pp. 2497–2506 (cit. on pp. 2, 13).

[10] Jean Alaux, Edouard Grave, Marco Cuturi, and Armand Joulin. "Unsupervised Hyperalignment for Multilingual Word Embeddings". *International Conference on Learning Representations (ICLR)* (2019) (cit. on pp. 2, 13, 15, 16, 36, 42, 50, 51).

[11] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. "A Neural Probabilistic Language Model". *Journal of Machine Learning Research*, vol. 3 (2003), pp. 1137–1155 (cit. on p. 2).

[12] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. "Natural language processing (almost) from scratch". *Journal of Machine Learning Research*, vol. 12, no. Aug (2011), pp. 2493–2537 (cit. on p. 2).

[13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient Estimation of Word Representations in Vector Space". In: *Proceedings of the 2nd International Conference on Learning Representation (ICLR)*. 2013 (cit. on p. 2).

[14] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014 (cit. on p. 2).

[15] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. "Allennlp: A deep semantic natural language processing platform". In: *Workshop for Natural Language Processing Open Source Software (NLP-OSS)*. July 2018, pp. 1–6 (cit. on p. 2).

[16] D Team et al. "Deeplearning4j: Open-source distributed deep learning for the jvm". *Apache Software Foundation License*, vol. 2 (2016) (cit. on p. 2).

[17] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. "FastText.zip: Compressing text classification models". *CoRR*, vol. abs/1612.03651 (2016) (cit. on p. 2).

[18] Zellig S Harris. "Distributional structure". *Word*, vol. 10, no. 2-3 (1954), pp. 146–162 (cit. on p. 3).

[19] Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. "Exploiting Similarities among Languages for Machine Translation". *CoRR*, vol. abs/1309.4168 (2013). arXiv: `1309.4168` (cit. on pp. 4, 5).

[20] Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. "Normalized Word Embedding and Orthogonal Transform for Bilingual Word Translation". In: *HLT-NAACL*. 2015 (cit. on pp. 4, 5).

[21] Peter H. Schönemann. "A Generalized Solution of the Orthogonal Procrustes problem". *Psychometrika*, vol. 31, no. 1 (Mar. 1966), pp. 1–10. ISSN: 1860-0980 (cit. on pp. 5, 34).

[22] David Alvarez-Melis and Tommi S Jaakkola. "Gromov-Wasserstein Alignment of Word Embedding Spaces". *arXiv preprint arXiv:1809.00013* (2018) (cit. on pp. 5, 9, 11, 36, 42, 50).

[23] Haggai Maron, Nadav Dym, Itay Kezurer, Shahar Kovalsky, and Yaron Lipman. "Point Registration via Efficient Convex Relaxation". *ACM Trans. Graph.* Vol. 35, no. 4 (July 2016), 73:1–73:12. ISSN: 0730-0301 (cit. on p. 6).

[24] Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam Kalai. "Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings". In: *Advances in neural information processing systems (NIPS)*. 2016 (cit. on p. 8).

[25] Marco Cuturi. "Sinkhorn Distances: Lightspeed Computation of Optimal Transport". In: *Advances in Neural Information Processing Systems (NIPS)*. 2013, pp. 2292–2300 (cit. on pp. 10, 26).

[26] Titouan Vayer, Laetita Chapel, Rémi Flamary, Romain Tavenard, and Nicolas Courty. "Fused Gromov-Wasserstein distance for structured objects: theoretical foundations and mathematical properties". *arXiv preprint arXiv:1811.02834* (2018) (cit. on p. 11).

[27] Gabriel Peyré, Marco Cuturi, and Justin Solomon. "Gromov-Wasserstein averaging of kernel and distance matrices". In: *International Conference on Machine Learning*. 2016, pp. 2664–2672 (cit. on pp. 11, 36–38).

[28] Hagai Taitelbaum, Gal Chechik, and Jacob Goldberger. "A Multi-Pairwise Extension of Procrustes Analysis for Multilingual Word Translation". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Nov. 2019, pp. 3558–3563 (cit. on pp. 13–15, 34, 42).

[29] Xilun Chen and Claire Cardie. "Unsupervised Multilingual Word Embeddings". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2018, pp. 261–270 (cit. on pp. 13, 15, 50).

[30] Martial Agueh and Guillaume Carlier. "Barycenters in the Wasserstein space". In: *Society for Industrial and Applied Mathematics (SIAM)*. Vol. 43. 2. 2011, pp. 904–924 (cit. on pp. 19, 20, 32, 33).

[31] Wilfrid Gangbo and Andrzej Święch. "Optimal Maps for the Multidimensional Monge-Kantorovich Problem". In: vol. 51. 1. Wiley Online Library, 1998, pp. 23–45 (cit. on p. 20).

[32] Gabriel Peyré, Marco Cuturi, et al. "Computational optimal transport". *Foundations and Trends® in Machine Learning*, vol. 11, no. 5-6 (2019), pp. 355–607 (cit. on p. 21).

[33] Marco Cuturi and Arnaud Doucet. "Fast Computation of Wasserstein Barycenters". In: *Proceedings of the 31st International Conference on Machine Learning (ICML)*. 2014, pp. 685–693 (cit. on pp. 22, 24, 29, 34).

[34] Jean-David Benamou and Guillaume Carlier and Marco Cuturi and Luca Nenna and Gabriel Peyré. “Iterative Bregman Projections for Regularized Transportation Problems”. In: 2015 (cit. on pp. 22, 24–26).

[35] Julien Rabin, Gabriel Peyré, Julie Delon, and Marc Bernot. “Wasserstein Barycenter and its Application to Texture Mixing”. In: *International Conference on Scale Space and Variational Methods in Computer Vision (SSVM)*. Springer. 2011, pp. 435–446 (cit. on pp. 22, 23).

[36] Justin Solomon, Fernando De Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. “Convolutional Wasserstein Distances: Efficient Optimal Transportation on Geometric Domains”. In: vol. 34. 4. ACM, 2015, p. 66 (cit. on pp. 22, 28).

[37] Lev M Bregman. “The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming”. In: vol. 3. 7. 1967, pp. 200–217 (cit. on p. 26).

[38] Heinz H Bauschke and Adrian S Lewis. “Dykstras algorithm with bregman projections: A convergence proof”. *Optimization*, vol. 48, no. 4 (2000), pp. 409–427 (cit. on p. 26).

[39] Sebastian Claici, Edward Chien, and Justin Solomon. “Stochastic Wasserstein Barycenters”. In: *International Conference on Machine Learning (ICML)*. Vol. 80. 2018 (cit. on pp. 30, 34).

[40] Gabaix Xavier. “Zipf’s Law for Cities: An Explanation”. *The Quarterly Journal of Economic*, vol. 114, no. 3 (1999), pp. 739–767 (cit. on p. 35).

[41] Goran Glavaš, Robert Litschko, Sebastian Ruder, and Ivan Vulić. “How to (Properly) Evaluate Cross-Lingual Word Embeddings: On Strong Baselines, Comparative Analyses, and Some Misconceptions”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, July 2019, pp. 710–721 (cit. on p. 41).

References

[1] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. "Distributed Representations of Words and Phrases and their Compositionality". In: *Neural Information Processing Systems (NIPS)*. 2013, pp. 3111–3119 (cit. on pp. 1, 2, 13).

[2] Reinhard Rapp. "Identifying Word Translations in Non-parallel Texts". In: *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*. ACL '95. Association for Computational Linguistics, 1995, pp. 320–322 (cit. on pp. 1, 3).

[3] Pascale Fung. "Compiling bilingual lexicon entries from a non-parallel English-Chinese corpus". In: *Third Workshop on Very Large Corpora*. 1995 (cit. on pp. 1, 3).

[4] Yuan Zhang, David Gaddy, Regina Barzilay, and Tommi Jaakkola. "Ten pairs to tag-Multilingual POS tagging via coarse mapping between embeddings". In: *Association for Computational Linguistics*. Association for Computational Linguistics. 2016 (cit. on p. 1).

[5] Georgiana Dinu and Marco Baroni. "Improving zero-shot learning by mitigating the hubness problem". In: *3rd International Conference on Learning Representations (ICLR)*. 2015 (cit. on pp. 1, 3, 8, 36, 41, 42).

[6] Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. "Offline Bilingual Word vectors, Orthogonal Transformations and the Inverted Softmax". In: 2017 (cit. on pp. 1, 13, 15, 16).

[7] Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. "Word Translation Without Parallel Data". In: *6th International Conference on Learning Representations (ICLR)*. Vol. abs/1710.04087. 2017 (cit. on pp. 2, 4, 5, 9, 12, 15, 16, 41, 42, 51).

[8] Hagai Taitelbaum, Gal Chechik, and Jacob Goldberger. "Multilingual Word Translation using Auxiliary Languages". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019 (cit. on pp. 2, 16, 17, 50).

[9] Ndapandula Nakashole and Raphael Flauger. "Knowledge Distillation for Bilingual Dictionary Induction". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Sept. 2017, pp. 2497–2506 (cit. on pp. 2, 13).

[10] Jean Alaux, Edouard Grave, Marco Cuturi, and Armand Joulin. "Unsupervised Hyperalignment for Multilingual Word Embeddings". *International Conference on Learning Representations (ICLR)* (2019) (cit. on pp. 2, 13, 15, 16, 36, 42, 50, 51).

[11] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. "A Neural Probabilistic Language Model". *Journal of Machine Learning Research*, vol. 3 (2003), pp. 1137–1155 (cit. on p. 2).

[12] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. "Natural language processing (almost) from scratch". *Journal of Machine Learning Research*, vol. 12, no. Aug (2011), pp. 2493–2537 (cit. on p. 2).

[13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient Estimation of Word Representations in Vector Space". In: *Proceedings of the 2nd International Conference on Learning Representation (ICLR)*. 2013 (cit. on p. 2).

[14] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014 (cit. on p. 2).

[15] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. "Allennlp: A deep semantic natural language processing platform". In: *Workshop for Natural Language Processing Open Source Software (NLP-OSS)*. July 2018, pp. 1–6 (cit. on p. 2).

[16] D Team et al. "Deeplearning4j: Open-source distributed deep learning for the jvm". *Apache Software Foundation License*, vol. 2 (2016) (cit. on p. 2).

[17] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. "FastText.zip: Compressing text classification models". *CoRR*, vol. abs/1612.03651 (2016) (cit. on p. 2).

[18] Zellig S Harris. "Distributional structure". *Word*, vol. 10, no. 2-3 (1954), pp. 146–162 (cit. on p. 3).

[19] Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. "Exploiting Similarities among Languages for Machine Translation". *CoRR*, vol. abs/1309.4168 (2013). arXiv: `1309.4168` (cit. on pp. 4, 5).

[20] Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. "Normalized Word Embedding and Orthogonal Transform for Bilingual Word Translation". In: *HLT-NAACL*. 2015 (cit. on pp. 4, 5).

[21] Peter H. Schönemann. "A Generalized Solution of the Orthogonal Procrustes problem". *Psychometrika*, vol. 31, no. 1 (Mar. 1966), pp. 1–10. ISSN: 1860-0980 (cit. on pp. 5, 34).

[22] David Alvarez-Melis and Tommi S Jaakkola. "Gromov-Wasserstein Alignment of Word Embedding Spaces". *arXiv preprint arXiv:1809.00013* (2018) (cit. on pp. 5, 9, 11, 36, 42, 50).

[23] Haggai Maron, Nadav Dym, Itay Kezurer, Shahar Kovalsky, and Yaron Lipman. "Point Registration via Efficient Convex Relaxation". *ACM Trans. Graph.* Vol. 35, no. 4 (July 2016), 73:1–73:12. ISSN: 0730-0301 (cit. on p. 6).

[24] Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam Kalai. "Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings". In: *Advances in neural information processing systems (NIPS)*. 2016 (cit. on p. 8).

[25] Marco Cuturi. "Sinkhorn Distances: Lightspeed Computation of Optimal Transport". In: *Advances in Neural Information Processing Systems (NIPS)*. 2013, pp. 2292–2300 (cit. on pp. 10, 26).

[26] Titouan Vayer, Laetita Chapel, Rémi Flamary, Romain Tavenard, and Nicolas Courty. "Fused Gromov-Wasserstein distance for structured objects: theoretical foundations and mathematical properties". *arXiv preprint arXiv:1811.02834* (2018) (cit. on p. 11).

[27] Gabriel Peyré, Marco Cuturi, and Justin Solomon. "Gromov-Wasserstein averaging of kernel and distance matrices". In: *International Conference on Machine Learning*. 2016, pp. 2664–2672 (cit. on pp. 11, 36–38).

[28] Hagai Taitelbaum, Gal Chechik, and Jacob Goldberger. "A Multi-Pairwise Extension of Procrustes Analysis for Multilingual Word Translation". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Nov. 2019, pp. 3558–3563 (cit. on pp. 13–15, 34, 42).

[29] Xilun Chen and Claire Cardie. "Unsupervised Multilingual Word Embeddings". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2018, pp. 261–270 (cit. on pp. 13, 15, 50).

[30] Martial Agueh and Guillaume Carlier. "Barycenters in the Wasserstein space". In: *Society for Industrial and Applied Mathematics (SIAM)*. Vol. 43. 2. 2011, pp. 904–924 (cit. on pp. 19, 20, 32, 33).

[31] Wilfrid Gangbo and Andrzej Święch. "Optimal Maps for the Multidimensional Monge-Kantorovich Problem". In: vol. 51. 1. Wiley Online Library, 1998, pp. 23–45 (cit. on p. 20).

[32] Gabriel Peyré, Marco Cuturi, et al. "Computational optimal transport". *Foundations and Trends® in Machine Learning*, vol. 11, no. 5-6 (2019), pp. 355–607 (cit. on p. 21).

[33] Marco Cuturi and Arnaud Doucet. "Fast Computation of Wasserstein Barycenters". In: *Proceedings of the 31st International Conference on Machine Learning (ICML)*. 2014, pp. 685–693 (cit. on pp. 22, 24, 29, 34).

[34] Jean-David Benamou and Guillaume Carlier and Marco Cuturi and Luca Nenna and Gabriel Peyré. “Iterative Bregman Projections for Regularized Transportation Problems”. In: 2015 (cit. on pp. 22, 24–26).

[35] Julien Rabin, Gabriel Peyré, Julie Delon, and Marc Bernot. “Wasserstein Barycenter and its Application to Texture Mixing”. In: *International Conference on Scale Space and Variational Methods in Computer Vision (SSVM)*. Springer. 2011, pp. 435–446 (cit. on pp. 22, 23).

[36] Justin Solomon, Fernando De Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. “Convolutional Wasserstein Distances: Efficient Optimal Transportation on Geometric Domains”. In: vol. 34. 4. ACM, 2015, p. 66 (cit. on pp. 22, 28).

[37] Lev M Bregman. “The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming”. In: vol. 3. 7. 1967, pp. 200–217 (cit. on p. 26).

[38] Heinz H Bauschke and Adrian S Lewis. “Dykstras algorithm with bregman projections: A convergence proof”. *Optimization*, vol. 48, no. 4 (2000), pp. 409–427 (cit. on p. 26).

[39] Sebastian Claici, Edward Chien, and Justin Solomon. “Stochastic Wasserstein Barycenters”. In: *International Conference on Machine Learning (ICML)*. Vol. 80. 2018 (cit. on pp. 30, 34).

[40] Gabaix Xavier. “Zipf’s Law for Cities: An Explanation”. *The Quarterly Journal of Economic*, vol. 114, no. 3 (1999), pp. 739–767 (cit. on p. 35).

[41] Goran Glavaš, Robert Litschko, Sebastian Ruder, and Ivan Vulić. “How to (Properly) Evaluate Cross-Lingual Word Embeddings: On Strong Baselines, Comparative Analyses, and Some Misconceptions”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, July 2019, pp. 710–721 (cit. on p. 41).