Bilingual Lexicon Induction for Low-Resource Languages using Graph Matching via Optimal Transport

Kelly Marchisio[1], Ali Saad-Eldin[3], Kevin Duh[1,4], Carey Priebe[2,4], Philipp Koehn[1]
Department of [1]Computer Science, [2]Department of Applied Mathematics and Statistics, [3]Department of Biomedical Engineering, [4]Human Language Technology Center of Excellence
Johns Hopkins University
{kmarc,asaadel1}@jhu.edu
kevinduh@cs.jhu.edu, {cep, phi}@jhu.edu

Abstract

Bilingual lexicons form a critical component of various natural language processing applications, including unsupervised and semisupervised machine translation and crosslingual information retrieval. We improve bilingual lexicon induction performance across 40 language pairs with a graph-matching method based on optimal transport. The method is especially strong with low amounts of supervision.

1 Introduction

Bilingual lexicon induction (BLI) from word embedding spaces is a popular task with a large body of existing literature (e.g. Mikolov et al., 2013; Artetxe et al., 2018; Conneau et al., 2018; Patra et al., 2019; Shi et al., 2021). The goal is to extract a dictionary of translation pairs given separate language-specific embedding spaces, which can then be used to bootstrap downstream tasks such as cross-lingual information retrieval and unsupervised/semi-supervised machine translation.

A great challenge across NLP is maintaining performance in low-resource scenarios. A common criticism of the BLI and low-resource MT literature is that while claims are made about diverse and under-resourced languages, research is often performed on down-sampled corpora of high-resource, highly-related languages on similar domains (Artetxe et al., 2020). Such corpora are not good proxies for true low-resource languages owing to data challenges such as dissimilar scripts, domain shift, noise, and lack of sufficient bitext (Marchisio et al., 2020). These differences can lead to dissimilarity between the embedding spaces (decreasing isometry), causing BLI to fail (Søgaard et al., 2018; Nakashole and Flauger, 2018; Ormazabal et al., 2019; Glavaš et al., 2019; Vulić et al., 2019; Patra et al., 2019; Marchisio et al., 2020).

There are two axes by which a language dataset is considered "low-resource". First, the language itself may be a low-resource language: one for which little bitext and/or monolingual text exists. Even for high-resource languages, the long tail of words may have poorly trained word embeddings due rarity in the dataset (Gong et al., 2018; Czarnowska et al., 2019). In the data-poor setting of true low-resource languages, a great majority of words have little representation in the corpus, resulting in poorly-trained embeddings for a large proportion of them. The second axis is low-supervision. Here, there are few ground-truth examples from which to learn. For BLI from word embedding spaces, low-supervision means there are few seeds from which to induce a relationship between spaces, regardless of the quality of the spaces themselves.

We bring a new algorithm for graph-matching based on optimal transport (OT) to the NLP and BLI literature. We evaluate using 40 language pairs under varying amounts of supervision. The method works strikingly well across language pairs, especially in low-supervision contexts. As low-supervision on low-resource languages reflects the real-world use case for BLI, this is an encouraging development on realistic scenarios.

2 Background

The typical baseline approach for BLI from word embedding spaces assumes that spaces can be mapped via linear transformation. Such methods typically involve solutions to the Procrustes problem (see Gower et al. (2004) for a review). Alternatively, a graph-based view considers words as nodes in undirected weighted graphs, where edges are the distance between words. Methods taking this view do not assume a linear mapping of the spaces exists, allowing for more flexible matching.

BLI from word embedding spaces Assume separately-trained monolingual word embedding spaces: $\mathbf{X} \in \mathbb{R}^{\mathbf{n\times d}}$, $\mathbf{Y} \in \mathbb{R}^{\mathbf{m\times d}}$ where n/m are the source/target language vocabulary sizes and d is the embedding dimension. We build the matrices $\overline{\mathbf{X}}$

and $\overline{\mathbf{Y}}$ of seeds from $\mathbf{X}$ and $\mathbf{Y}$, respectively, such that given s seed pairs $(x_1, y_1), (x_2, y_2), ...(x_s, y_s)$, the first row of $\overline{\mathbf{X}}$ is x_1, the second row is x_2, etc. We build $\overline{\mathbf{Y}}$ analogously for the y-component of each seed pair. The goal is to recover matches for the $\mathbf{X} \setminus \overline{\mathbf{X}}$ and/or $\mathbf{Y} \setminus \overline{\mathbf{Y}}$ non-seed words.

Procrustes Many BLI methods use solutions to the Procrustes problem (e.g. Artetxe et al., 2019b; Conneau et al., 2018; Patra et al., 2019). These compute the optimal transform $\mathbf{W}$ to map seeds:

$$\min_{\mathbf{W} \in \mathbb{R}^{\mathbf{d} \times \mathbf{d}}} ||\overline{\mathbf{X}}\mathbf{W} - \overline{\mathbf{Y}}||_{\mathbf{F}}^{\mathbf{2}} \tag{1}$$

Once solved for $\mathbf{W}$, then $\mathbf{XW}$ and $\mathbf{Y}$ live in a shared space and translation pairs can be extracted via nearest-neighbor search. Constrained to the space of orthogonal matrices, Equation 1 has a simple closed-form solution (Schönemann, 1966):

$$\mathbf{W} = \mathbf{VU^T} \quad U\Sigma V = \text{SVD}(\overline{\mathbf{Y}}^T\overline{\mathbf{X}})$$

Graph View Here, words are nodes in monolingual graphs $\mathbf{G_x}, \mathbf{G_y} \in \mathbb{R}^{\mathbf{n} \times \mathbf{n}}$, and cells in $\mathbf{G_x}, \mathbf{G_y}$ are edge weights representing distance between words. As is common in NLP, we use cosine similarity. The objective function is Equation 2, where Π is the set of ***permutation matrices***.[1] Intuitively, $\mathbf{PG_yP^T}$ finds the optimal relabeling of $\mathbf{G_y}$ to align with $\mathbf{G_x}$. This "minimizes edge-disagreements" between $\mathbf{G_x}$ and $\mathbf{G_y}$. This graph-matching objective is NP-Hard. Equation 3 is equivalent.

$$\min_{\mathbf{P} \in \mathbf{\Pi}} ||\mathbf{G_x} - \mathbf{PG_yP^T}||_{\mathbf{F}}^{\mathbf{2}} \tag{2}$$

$$\max_{\mathbf{P} \in \mathbf{\Pi}} \textit{trace}(\mathbf{G_x}^\mathbf{T}\mathbf{PG_yP^T}) \tag{3}$$

Ex. Take source words x_1, x_2. We wish to recover valid translations y_{x_1}, y_{x_2}. If distance$(x_1, x_2) =$ distance(y_{x_1}, y_{x_2}), a solution P can have an edge-disagreement of 0 here. We then extract y_{x_1}, y_{x_2} as translations of x_1, x_2. In reality, though, it is unlikely that distance(x_1, x_2) = distance(y_{x_1}, y_{x_2}). Because Equation 2 finds the ideal P to minimize edge disagreements over the entire graphs, we hope that nodes paired by P are valid translations. If $\mathbf{G_x}$ and $\mathbf{G_y}$ are isomorphic and there is a unique solution, then P correctly recovers all translations.

Graph-matching is an active research field and is computationally prohibitive on large graphs, but approximation algorithms exist. BLI involves matching large, non-isomorphic graphs—among the greatest challenges for graph-matching.

2.1 FAQ Algorithm for Graph Matching

Vogelstein et al. (2015)'s Fast Approximate Quadratic Assignment Problem algorithm (FAQ) uses gradient ascent to approximate a solution to Equation 2. Motivated by "connectonomics" in neuroscience (the study of brain graphs with biological [groups of] neurons as nodes and neuronal connections as edges), FAQ was designed to perform accurately and efficiently on large graphs.

FAQ relaxes the search space of Equation 3 to allow any doubly-stochastic matrix (the set $\mathcal{D}$). Each cell in a doubly-stochastic matrix is a non-negative real number and each row/column sums to 1. The set $\mathcal{D}$ thus contains Π but is much larger. Relaxing the search space makes it easier to optimize Equation 3 via gradient ascent/descent.[2] FAQ solves the objective with the Frank-Wolfe method (Frank et al., 1956) then projects back to a permutation matrix.

Algorithm 1 is FAQ; $f(P)$ = *trace*$(\mathbf{G_x}^\mathbf{T}\mathbf{PG_yP^T})$. These may be built as $\mathbf{G_x} = \mathbf{XX^T}$ and $\mathbf{G_y} = \mathbf{YY^T}$. $\mathbf{G_x}$ and $\mathbf{G_y}$ need not have the same dimensionality. Step 2 finds a permutation matrix approximation $Q^{\{i\}}$ to $P^{\{i\}}$ in the direction of the gradient. Finding such a P requires approximation when P is high-dimensional. Here, it is solved via the **Hungarian Algorithm** (Kuhn, 1955; Jonker and Volgenant, 1987), whose solution is a permutation matrix. Finally, P^n is projected back onto to the space of permutation matrices. Seeded Graph Matching (SGM; Fishkind et al., 2019) is a variant of FAQ allowing for supervision, and was recently shown to be effective for BLI by Marchisio et al. (2021). The interested reader may find Vogelstein et al. (2015) and Fishkind et al. (2019) enlightening for descriptive intuitions of the FAQ and SGM algorithms.[3]

Strengths/Weaknesses FAQ/SGM perform well solving the exact graph-matching problem: where graphs are isomorphic and a full matching exists. In reality, large graphs are rarely isomorphic. For BLI, languages have differing vocabulary size, syn-

[1] A permutation matrix represents a one-to-one mapping: There is a single 1 in each row and column, and 0 elsewhere.

[2] "descent" for the Quadratic Assignment Problem, "ascent" for the Graph Matching Problem. The optimization objectives are equivalent: See Vogelstein et al. (2015) for a proof.

[3] 'Section 3: Fast Approximate QAP Algorithm' (Vogelstein et al., 2015), 'Section 2.2. From FAQ to SGM' (Fishkind et al., 2019).

Algorithm 1 FAQ Algorithm for Graph Matching

```
Let: $\mathbf{G_x}, \mathbf{G_y} \in \mathbb{R}^{\mathbf{n \times n}}$, $P^{\{0\}} \in \mathcal{D}$ (dbl-stoch.)
while stopping criterion not met do
    1. Calculate $\nabla f(P^{\{i\}})$:
       $\nabla f(P^{\{i\}}) = G_x P^{\{i\}} G_y^T + G_x^T P^{\{i\}} G_y$
    2. $Q^{\{i\}}$ = permutation matrix approx. to $\nabla f(P^{\{i\}})$
       via Hungarian Algorithm
    3. Calculate step size:
       $\arg\max_{\alpha \in [0,1]} f(\alpha P^{\{i\}} + (1-\alpha) Q^{\{i\}})$
    4. Update $P^{\{i+1\}} := \alpha P^{\{i\}} + (1-\alpha) Q^{\{i\}}$
end while
return permutation matrix approx. to $P^{\{n\}}$ via Hung. Alg.
```

onyms/antonyms, and idiosyncratic concepts; it is more natural to assume that an exact matching between word spaces does *not* exist, and that multiple matchings may be equally valid. This is an inexact graph-matching problem. FAQ generally performs poorly finding non-seeded inexact matchings (Saad-Eldin et al., 2021).

2.2 GOAT

Graph Matching via OptimAl Transport (GOAT) (Saad-Eldin et al., 2021) is a new graph-matching method which uses advances in OT. Similar to SGM, GOAT amends FAQ and can use seeds. GOAT has been successful for the inexact graph-matching problem on non-isomorphic graphs: whereas FAQ rapidly fails on non-isomorphic graphs, GOAT maintains strong performance.

Optimal Transport OT is an optimization problem concerned with the most efficient way to transfer probability mass from distribution μ to distribution v. Discrete[4] OT minimizes the inner product of a transportation "plan" matrix P with a cost matrix C, as in Equation 4. $\langle\cdot,\cdot\rangle$ is the Frobenius inner product.

$$P^* = \arg\min_{P \in \mathcal{U}(r,c)} \langle P, C \rangle \tag{4}$$

P is an element of the "transportation polytope" $U(r,c)$—the set of matrices whose rows sum to r and columns sum to c. The Hungarian Algorithm approximately solves OT, but the search space is restricted to permutation matrices.

Sinkhorn: Lightspeed OT Cuturi (2013) introduce Sinkhorn distance, an approximation of OT distance that can be solved quickly and accurately by adding an entropy penalty h to Equation 4. Adding h makes the objective easier and more efficient to compute, and encourages "intermediary" solutions similar to that seen in the **Intuition** subsection.

$$P^\lambda = \arg\min_{P \in \mathcal{U}(r,c)} \langle P, C \rangle - \frac{1}{\lambda} h(P) \tag{5}$$

As $\lambda \to \infty$, P^λ approaches the ideal transportation matrix P^*. Cuturi (2013) show that Equation 5 can be computed using Sinkhorn's algorithm (Sinkhorn, 1967). The interested reader can see details of the algorithm in Cuturi (2013); Peyre and Cuturi (2019). Unlike the Hungarian Algorithm, Sinkhorn has no restriction to a permutation matrix solution and can be solved over any $U(r,c)$.

Sinkhorn in GOAT GOAT uses Cuturi (2013)'s algorithm to solve Equation 5 over $U(1,1)$, the set of doubly-stochastic matrices $\mathcal{D}$. They call this the "doubly stochastic OT problem", and the algorithm that solves it "Lightspeed Optimal Transport" (LOT). Although Sinkhorn distance was created for efficiency, Saad-Eldin et al. (2021) find that using the matrix P^λ that minimizes Sinkhorn distance also improves matching performance on large and non-isometric graphs. Algorithm 2 is GOAT.

Algorithm 2 GOAT

```
Let: $\mathbf{G_x}, \mathbf{G_y} \in \mathbb{R}^{\mathbf{n \times n}}$, $P^{\{0\}} \in \mathcal{D}$ (dbl-stoch.)
while stopping criterion not met do
    1. Calculate $\nabla f(P^{\{i\}})$:
       $\nabla f(P^{\{i\}}) = G_x P^{\{i\}} G_y^T + G_x^T P^{\{i\}} G_y$
    2. $Q^{\{i\}}$ = dbl-stoch. approx. to $\nabla f(P^{\{i\}})$ via LOT.
    3. Calculate step size:
       $\arg\max_{\alpha \in [0,1]} f(\alpha P^{\{i\}} + (1-\alpha) Q^{\{i\}})$
    4. Update $P^{\{i+1\}} := \alpha P^{\{i\}} + (1-\alpha) Q^{\{i\}}$
end while
return permutation matrix approx. to $P^{\{n\}}$ via Hung. Alg.
```

Intuition The critical difference between SGM/FAQ and GOAT is how each calculates step direction based on the gradient. Under the hood, each algorithm maximizes $trace(Q^T \nabla f(P^{\{i\}})$ to compute $Q^{\{i\}}$ (the step direction) in Step 2 of their respective algorithms. See Saad-Eldin et al. (2021) or Fishkind et al. (2019) for a derivation. FAQ uses the Hungarian Algorithm and GOAT uses LOT.

For $\nabla f(P^{\{i\}})$ below, there are *two* valid permutation matrices Q_1 and Q_2 that maximize the trace. When multiple solutions exist, the Hungarian Algorithm chooses one arbitrarily. Thus, updates of P

[4]As ours is, as we compute over matrices.

in FAQ are constrained to be permutation matrices.

$$\nabla f(P^{\{i\}}) = \begin{pmatrix} 0 & 3 & 0 \\ 2 & 1 & 2 \\ 0 & 0 & 0 \end{pmatrix}$$

$$Q_1 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \quad Q_2 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\mathit{trace}(Q_1^T \nabla f(P^{\{i\}})) = \mathit{trace}(Q_2^T \nabla f(P^{\{i\}})) = 5$$

Concerningly, Saad-Eldin et al. (2021) find that seed order influences the solution in a popular implementation of the Hungarian Algorithm. Since BLI is a high-dimensional many-to-many task, arbitrary choices could meaningfully affect the result.

GOAT, on the other hand, can step in the direction of a doubly-stochastic matrix. Saad-Eldin et al. (2021) prove that given multiple permutation matrices that equally approximate the gradient at $P^{\{i\}}$, any convex linear combination is a doubly stochastic matrix that equally approximates the gradient:

$$P_\lambda = \sum_i^n \lambda_i P_i \quad s.t.\ \lambda_1 + ... + \lambda_n = 1;\ \lambda_i \in [0, 1]$$

P_λ is a weighted combination of *many* valid solutions—obviating the need to arbitrarily select one for the gradient-based update. LOT's output of a doubly-stochastic matrix in Step 2 is similar to finding a P_λ in that it needn't discretize to a single permutation matrix. In this way, GOAT can be thought of as taking a step that incorporates many possible permutation solutions. For instance, GOAT may select $Q_{ds} = \frac{1}{2}Q_1 + \frac{1}{2}Q_2$, which also maximizes $\mathit{trace}(Q^T \nabla f(P^{\{i\}})$.

$$Q_{ds} = \begin{pmatrix} 0 & 1 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix}$$

$$\mathit{trace}(Q_{ds}^T \nabla f(P^{\{i\}})) = 5$$

Thus whereas FAQ takes non-deterministic "choppy" update steps, GOAT optimizes smoothly and deterministically. Figure 1 is an illustration.

3 Experimental Setup

We run Procrustes, SGM, and GOAT on 40 language pairs. We also run system combination experiments similar to Marchisio et al. (2021). We evaluate with the standard precision@1 (P@1).

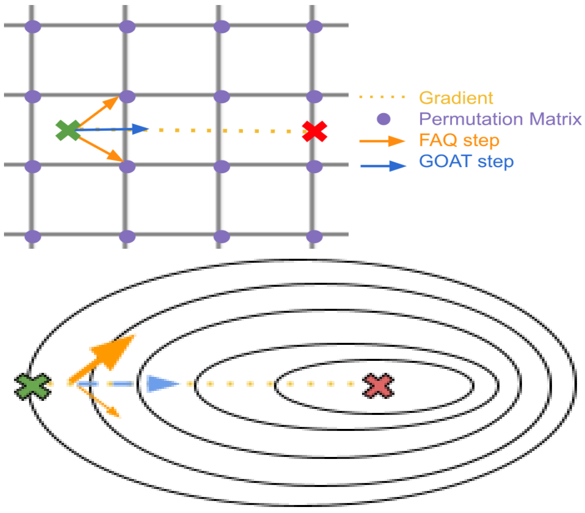

Figure 1: Optimization step of FAQ vs. GOAT. FAQ arbitrarily chooses the direction of a permutation matrix. GOAT averages permutation matrices to take a smoother path.

We induce lexicons using **(1)** the closed-form solution to the orthogonal Procrustes problem of Equation 1, extracting nearest neighbors using CSLS (Conneau et al., 2018), **(2)** SGM, solving the seeded version of Equation 2, and **(3)** GOAT. Word graphs are $\mathbf{G_x} = \mathbf{XX^T}$, $\mathbf{G_y} = \mathbf{YY^T}$.

System Combination We perform system combination experiments analogous to those of Marchisio et al. (2021), incorporating GOAT. Figure 2 shows the system, which is made of two components: GOAT run in forward and reverse directions, and "Iterative Procrustes with Stochastic-Add" from Marchisio et al. (2021). This iterative version of Procrustes runs Procrustes in source→target and target→source directions and feeds H random hypotheses from the intersection of both directions into another run of Procrustes with the gold seeds. The process repeats for I iterations, adding H more random hypotheses each time until all are chosen. We set $H = 100$ and $I = 5$, as in the original work.

3.1 Data & Software

We use publicly-available fastText word embeddings (Bojanowski et al., 2017)[5] which we normalize, mean-center, and renormalize (Artetxe et al., 2018; Zhang et al., 2019) and bilingual dictionaries from MUSE[6] filtered to be one-to-one.[7] For languages with 200,000+ embeddings, we use the first

[5]https://fasttext.cc/docs/en/pretrained-vectors.html

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

[7]For each source word, keep the first unused target word. Targets are in arbitrary order, so this is random sampling.

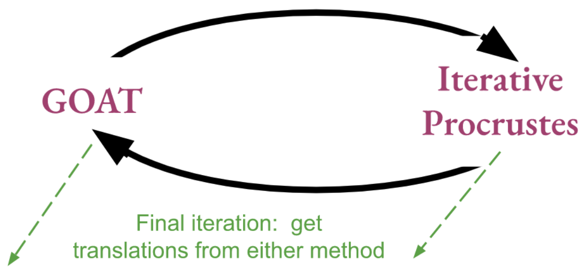

Figure 2: Combination system: Iterative Procrustes (IterProc) & GOAT. (1) run GOAT in forward/reverse directions (2) intersect hypotheses, pass to IterProc, (3) run IterProc forward & reverse (4) intersect hypotheses, pass to Step (1). Repeat N cycles. (End) Get final translations from forward run of GOAT or IterProc.

200,000. Dictionary and embeddings space sizes are in Appendix Table A1. Each language pair has ∼4100-4900 translation pairs post-filtering. We choose 0-4000 pairs in frequency order as seeds for experiments, leaving the rest as the test set.[8] For SGM and GOAT, we use the publicly-available implementations from the GOAT repository[9] with default hyperparameters (barycenter initialization). We set reg=500 for GOAT. For system combination experiments, we amend the code from Marchisio et al. (2021)[10] to incorporate GOAT.

3.2 Languages

The languages chosen reflect various language families and writing systems. The language families represented are:

- **Austroasiatic**: *Vietnamese*
- **Austronesian**: *Indonesian, Malay*
- **Dravidian**: *Tamil*
- **Japonic**: *Japanese*
- **Sino-Tibetan**: *Chinese*
- **Uralic**: *Estonian*
- **Indo-European**
  - **Balto-Slavic**: *Macedonian, Bosnian, Russian*
  - **Germanic**: *English, German*
  - **Indo-Iranian**: *Bengali, Persian*
  - **Romance**: *French, Spanish, Portuguese, Italian*

The chosen languages use varying writing systems, including those using or derived from Latin, Cyrillic, Arabic, Tamil, and character-based scripts.

[8] Ex. En-De with 100 seeds has 4803 test items. With 1000 seeds, the test set contains 3903 items.

[9] https://github.com/neurodata/goat. Some exps. used SGM from Graspologic (github.com/microsoft/graspologic; Chung et al., 2019), but they are mathematically equal.

[10] https://github.com/kellymarchisio/euc-v-graph-bli

4 Results

Results of Procrustes vs. SGM vs. GOAT are in Table 1, visualized in Figure 3.

Procrustes vs. SGM Marchisio et al. (2021) conclude that SGM strongly outperforms Procrustes for English→German and Russian→English with 100+ seeds. We find that the trend holds across language pairs, with the effect even stronger with less supervision. SGM performs reasonably with only 50 seeds for nearly all languages, and with only 25 seeds in many. Chinese↔English and Japanese↔English perform relatively worse, and highly-related languages perform best: French, Spanish, Italian, and Portuguese. German↔English performance is low relative to some less-related languages, which have surprisingly strong performance from SGM: Indonesian↔English and Macedonian↔English score $P@1 \approx 50$-60, even with low supervision. Except for the aforementioned highly-related language pairs, Procrustes does not perform above ~ 10 for any language pair with $\leq$ 100 seeds, whereas SGM exceeds $P@1 = 10$ with only 25 seeds for 33 of 40 pairs.

SGM vs. GOAT GOAT improves considerably over SGM for nearly all language pairs, and the effect is particularly strong with very low amounts of seeds and less-related languages. GOAT improves upon SGM by +19.0, +8.5, and +7.9 on English→Bengali with 25, 50, and 75 seeds, respectively. As the major use case of low-resource BLI and MT is dissimilar languages with low supervision, this is an encouraging result. It generally takes 200+ seeds for SGM to achieve similar scores to GOAT with just 25 seeds.

4.1 Isomorphism of Embedding Spaces

Eigenvector similarity (EVS; Søgaard et al., 2018) measures isomorphism of embedding spaces based on the difference of Laplacian eigenvalues. Gromov-Hausdorff distance (GH) measures distance based on nearest neighbors after an optimal orthogonal transformation (Patra et al., 2019). EVS and GH are symmetric, and lower means more isometric spaces. Refer to the original papers for mathematical descriptions. We compute the metrics over the word embedding using scripts from Vulić et al. (2020)[11] and show results in Table 2. We observe a

[11] https://github.com/cambridgeltl/iso-study/scripts

Seeds	P	S	G	Δ	P	S	G	Δ	P	S	G	Δ	P	S	G	Δ
	English-to-*															
	Bengali				Bosnian				Estonian				Persian			
25	0.0	12.9	**31.9**	(+19.0)	0.1	37.0	**48.1**	(+11.1)	0.1	37.5	**47.0**	(+9.5)	0.2	35.2	**42.7**	(+7.5)
200	4.4	30.9	**35.4**	(+4.5)	7.0	44.5	**49.9**	(+5.4)	9.0	46.0	**49.6**	(+3.6)	7.0	40.0	**44.4**	(+4.4)
1000	30.6	40.7	**41.3**	(+0.6)	39.1	54.4	**55.1**	(+0.7)	45.5	57.0	**57.3**	(+0.3)	40.9	48.2	**49.9**	(+1.7)
2000	36.9	**45.9**	45.5	(-0.4)	45.9	**58.1**	57.1	(-1.0)	53.1	63.4	**64.0**	(+0.6)	47.6	54.1	**54.7**	(+0.6)
	Indonesian				Macedonian				Malay				Tamil			
25	0.1	49.2	**51.3**	(+2.1)	0.1	51.0	**55.8**	(+4.8)	0.3	24.9	**36.6**	(+11.7)	0.1	1.5	**1.8**	(+0.3)
200	13.0	54.1	**55.9**	(+1.8)	12.2	53.3	**57.6**	(+4.3)	11.5	43.3	**46.0**	(+2.7)	4.1	26.7	**31.0**	(+4.3)
1000	58.0	**64.5**	63.6	(-0.9)	51.0	60.3	**61.8**	(+1.5)	48.9	**58.9**	58.3	(-0.6)	26.9	**36.2**	**36.2**	(+0.0)
2000	65.0	**70.7**	69.5	(-1.2)	56.3	63.9	**64.6**	(+0.7)	55.3	**65.0**	63.0	(-2.0)	32.3	**40.6**	39.8	(-0.8)
	Vietnamese				Chinese				Japanese				Russian			
25	0.2	0.4	**0.4**	(+0.0)	0.3	**8.7**	7.9	(-0.8)	0.0	**1.1**	1.0	(-0.1)	0.4	49.6	**55.7**	(+6.1)
200	2.9	34.8	**40.6**	(+5.8)	12.0	**22.7**	17.3	(-5.4)	8.3	**20.7**	15.6	(-5.1)	16.5	54.3	**57.3**	(+3.0)
1000	37.4	53.7	**54.9**	(+1.2)	**44.4**	34.5	29.5	(-5.0)	**47.8**	35.6	27.4	(-8.2)	58.3	**61.7**	61.5	(-0.2)
2000	50.7	60.7	**61.1**	(+0.4)	**49.3**	45.6	41.7	(-3.9)	**56.0**	45.1	40.3	(-4.8)	65.8	67.4	**67.5**	(+0.1)
	German				French				Spanish				Italian			
25	0.3	44.8	**48.5**	(+3.7)	0.4	58.6	**62.4**	(+3.8)	0.3	59.5	**62.8**	(+3.3)	0.4	60.0	**63.0**	(+3.0)
200	16.1	47.5	**50.2**	(+2.7)	24.9	60.9	**63.2**	(+2.3)	27.2	62.1	**63.9**	(+1.8)	24.9	63.2	**64.4**	(+1.2)
1000	**57.2**	54.9	54.5	(-0.4)	67.9	**68.3**	67.9	(-0.4)	**69.6**	68.8	69.0	(+0.2)	69.8	**70.5**	70.3	(-0.2)
2000	**63.1**	61.4	61.8	(+0.4)	72.5	**72.9**	72.2	(-0.7)	74.1	**75.2**	74.9	(-0.3)	75.6	75.9	**76.0**	(+0.1)
	*-to-English															
	Bengali				Bosnian				Estonian				Persian			
25	0.2	37.2	**44.4**	(+7.2)	0.2	44.7	**54.6**	(+9.9)	0.3	55.9	**63.2**	(+7.3)	0.1	37.1	**45.1**	(+8.0)
200	7.6	43.0	**46.6**	(+3.6)	6.8	50.4	**56.2**	(+5.8)	12.6	60.7	**64.8**	(+4.1)	9.9	42.7	**46.8**	(+4.1)
1000	39.0	**51.5**	51.3	(-0.2)	44.7	59.2	**61.6**	(+2.4)	54.6	67.7	**70.0**	(+2.3)	45.9	48.5	**49.6**	(+1.1)
2000	45.2	**55.2**	54.2	(-1.0)	54.5	64.8	**65.9**	(+1.1)	62.6	72.8	**74.2**	(+1.4)	50.3	**53.8**	53.6	(-0.2)
	Indonesian				Macedonian				Malay				Tamil			
25	0.1	54.6	**58.0**	(+3.4)	0.2	60.1	**63.2**	(+3.1)	0.2	10.0	**38.6**	(+28.6)	0.2	35.2	**44.4**	(+9.2)
200	13.3	55.7	**58.3**	(+2.6)	16.5	62.0	**64.5**	(+2.5)	15.5	56.2	**58.9**	(+2.7)	7.4	43.4	**45.8**	(+2.4)
1000	58.5	63.2	**64.1**	(+0.9)	58.5	66.1	**67.9**	(+1.8)	55.7	61.6	**62.3**	(+0.7)	37.1	49.6	**50.3**	(+0.7)
2000	66.0	**70.1**	**70.1**	(+0.0)	62.6	**71.7**	71.6	(-0.1)	60.2	**67.4**	67.2	(-0.2)	45.2	**55.1**	53.5	(-1.6)
	Vietnamese				Chinese				Japanese				Russian			
25	0.1	1.0	**3.3**	(+2.3)	0.1	6.8	**9.3**	(+2.5)	0.2	8.0	**31.2**	(+23.2)	0.3	49.1	**54.4**	(+5.3)
200	1.7	41.3	**48.6**	(+7.3)	12.0	**19.8**	16.1	(-3.7)	19.1	**34.6**	34.2	(-0.4)	16.6	52.5	**55.2**	(+2.7)
1000	29.1	52.4	**57.2**	(+4.8)	**44.5**	33.2	31.9	(-1.3)	**47.7**	42.7	38.9	(-3.8)	56.6	58.1	**60.3**	(+2.2)
2000	56.3	64.1	**66.3**	(+2.2)	**50.8**	41.5	41.8	(+0.3)	**54.0**	48.6	47.6	(-1.0)	62.7	67.1	**68.5**	(+1.4)
	German				French				Spanish				Italian			
25	0.2	30.3	**34.1**	(+3.8)	0.3	62.5	**65.3**	(+2.8)	0.4	61.4	**64.4**	(+3.0)	0.4	63.6	**66.9**	(+3.3)
200	10.6	45.0	**48.3**	(+3.3)	26.8	64.2	**66.4**	(+2.2)	32.6	65.0	**65.9**	(+0.9)	28.2	66.6	**68.7**	(+2.1)
1000	52.8	53.1	**53.3**	(+0.2)	**71.4**	70.1	70.4	(+0.3)	69.5	69.8	**70.6**	(+0.8)	71.6	71.8	**73.0**	(+1.2)
2000	**60.7**	60.5	60.4	(-0.1)	**76.2**	75.3	74.8	(-0.5)	72.9	**74.5**	73.9	(-0.6)	**76.3**	75.6	75.8	(+0.2)
	-to-															
	German-Spanish				Italian-French				Spanish-Portuguese				Portuguese-German			
25	0.5	67.4	**70.5**	(+3.1)	1.1	85.4	**87.3**	(+1.9)	2.5	94.3	**94.9**	(+0.6)	0.5	72.5	**76.2**	(+3.7)
200	26.0	67.9	**71.0**	(+3.1)	57.3	86.7	**87.7**	(+1.0)	74.1	94.9	**95.2**	(+0.3)	36.3	74.8	**76.9**	(+2.1)
1000	71.1	74.0	**75.4**	(+1.4)	86.1	89.1	**89.4**	(+0.3)	93.3	95.8	**96.0**	(+0.2)	75.5	78.9	**79.6**	(+0.7)
2000	76.1	78.8	**79.1**	(+0.3)	88.8	89.6	**90.4**	(+0.8)	94.5	97	**97.1**	(+0.1)	80	81.9	**82.0**	(+0.1)
	Spanish-German				French-Italian				Portuguese-Spanish				German-Portuguese			
25	0.5	62.1	**66.2**	(+4.1)	0.6	88.9	**90.2**	(+1.3)	1.1	89.7	**90.5**	(+0.8)	0.3	72.9	**76.2**	(+3.3)
200	27.7	65.2	**67.2**	(+2.0)	58.4	90.4	**91.2**	(+0.8)	70.0	90.4	**90.7**	(+0.3)	24.3	73.9	**77.1**	(+3.2)
1000	68.8	70.6	**70.6**	(+0.0)	89.6	92.1	**92.7**	(+0.6)	89.5	90.4	**91.1**	(+0.7)	73.7	79.8	**80.8**	(+1.0)
2000	73.6	74.0	**74.0**	(+0.0)	91.8	93.9	**94.1**	(+0.2)	90.9	91.7	**92.3**	(+0.6)	78.7	80.6	**82.2**	(+1.6)
	Averages															
	English-to-*				*-to-English				Non-English				Overall			
25	0.2	33.2	**38.6**	(+5.4)	0.2	38.6	**46.3**	(+7.7)	0.9	79.2	**81.5**	(+2.3)	0.3	37.1	**41.9**	(+4.8)
200	12.6	44.1	**46.4**	(+2.3)	14.8	50.2	**52.8**	(+2.6)	23.4	40.3	**41.1**	(+0.8)	16.9	44.8	**46.8**	(+2.0)
1000	49.6	**54.3**	53.7	(-0.6)	52.3	57.4	**58.3**	(+0.9)	40.5	41.9	**42.2**	(+0.3)	47.5	51.2	**51.4**	(+0.2)
2000	56.2	**60.4**	59.6	(-0.8)	59.8	63.6	**63.7**	(+0.1)	42.1	43.0	**43.2**	(+0.2)	52.7	**55.7**	55.5	(-0.2)

Table 1: P@1 of Procrustes (P), SGM (S) or GOAT (G). Δ is gain/loss of GOAT vs. SGM. Full results in Appendix. Figure 3 is a visualization of these results.

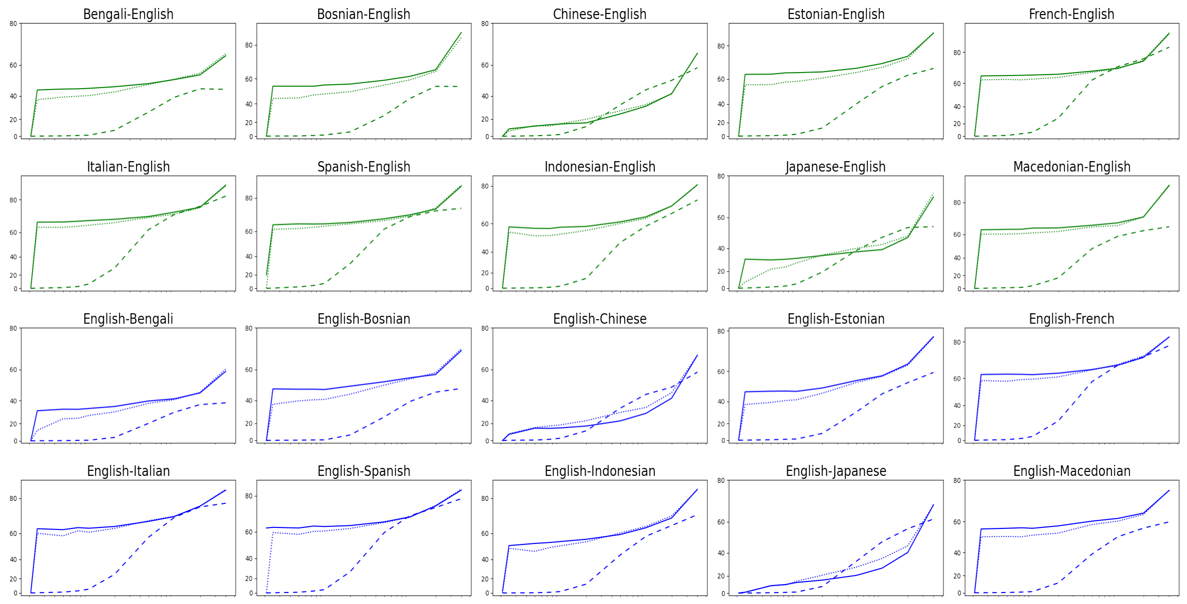

Figure 3: Visualization of Table 1 for select languages. Procrustes (dashed) vs. SGM (dotted) vs. GOAT (solid). X-axis: # of seeds (log scale). Y-axis: Precision@1 (↑ is better). GOAT is typically best.

	EVS	GH		EVS	GH
bn	37.79	0.49	it	22.42	0.20
bs	35.93	0.41	ja	894.20	0.55
de	11.49	0.31	mk	151.02	0.19
es	9.91	0.21	ms	153.42	0.49
et	35.22	0.68	ru	14.19	0.46
fa	86.98	0.39	ta	56.66	0.26
fr	27.92	0.17	vi	256.28	0.42
id	188.98	0.39	zh	519.82	0.61

Table 2: Degree of isomorphism of embedding spaces in relation to English. EVS = Eigenvector Similarity. GH = Gromov-Hausdorff Distance. ↓: more isomorphic.

moderate correlation between EVS and GH (Spearman's $\rho = 0.434$, Pearson's $r = 0.44$).

Figure 4 shows the relationship between relative isomorphism of each language vs. English, and performance of Procrustes/GOAT at 200 seeds. Trends indicate that higher isomorphism varies with higher precision from Procrustes and GOAT. GH shows a moderate to strong negative Pearson's correlation with performance from Procrustes and GOAT: $r = -0.47$ and $r = -0.53$, respectively, for *-to-en and -0.55 and -0.61 for en-to-*. EVS correlates weakly negatively with performance from Procrustes (*-to-en: -0.06, en-to-*: -0.28) and strongly negatively with GOAT (*-to-en: -0.67, en-to-*: -0.75). As higher GH/EVS indicates less isomorphism, negative correlations imply that lower de-

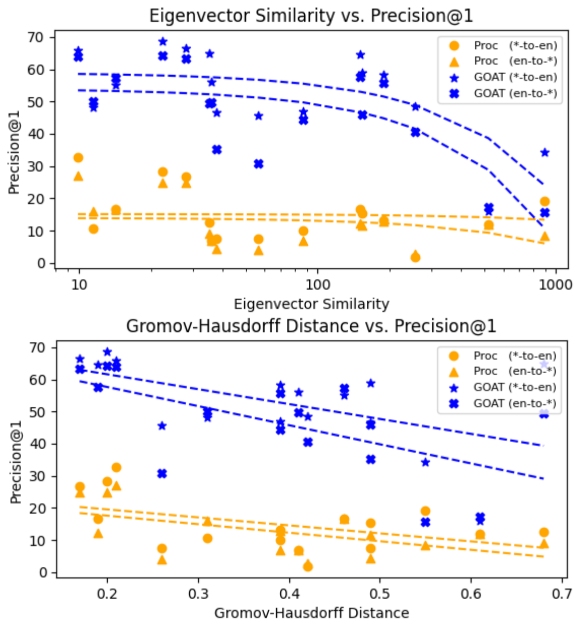

Figure 4: X-axis: Eigenvector Similarity (EVS) / Gromov-Hausdorff (GH) Distance of language compared to English. Y-axis: Precision@1 from Procrustes & GOAT with 200 seeds. ↓ EVS/GH = ↑ isomorphic.

grees of isomorphism correlate with lower scores from Procrustes/GOAT.

4.2 System Combination

System combination results are in Table 3. Similar to Marchisio et al. (2021)'s findings for their combined Procrustes/SGM system, we find (1) our combined Procrustes/GOAT system outperforms

Procrustes and GOAT alone, (2) ending with the Iterative Procrustes is best for moderate amounts of seeds, (3) ending with GOAT is best for very low or very high number of seeds.

Whether we end with Iterative Procrustes vs. GOAT is critically important for the lowest seed sizes: -EndGOAT (-EG) usually fails with 25 seeds; all language pairs except German↔English and Russian↔English score $P@1 < 15.0$, and most score $P@1 < 2.0$. Simply switching the order of processing in the combination system, however, boosts performance dramatically: ex. from 0.6 for StartProc-EndGOAT to 61.5 for StartGOAT-EndProc for Bosnian→English with 25 seeds.

There are some language pairs such as English→Persian and Russian↔English where a previous experiment with no seeds had reasonable performance, but the combined system failed. It is worth investigating where this discrepancy arises.

5 Discussion

We have seen GOAT's strength in low-resource scenarios and in non-isomorphic embedding spaces. As the major use case of low-resource BLI and MT is dissimilar languages with low supervision, GOAT's strong performance is an encouraging result for real-world applications. Furthermore, GOAT outperforms SGM. As the graph-matching objective is NP-hard so all algorithms are approximate, GOAT does a better job by making a better calculation of step direction. Chinese↔English and Japanese↔English are outliers, which is worthy of future investigation. Notably, these languages have very poor isomorphism scores in relation to English.

Why might graph-based methods work? The goal for Procrustes is to find the ideal linear transformation $W_{ideal} \in \mathbb{R}^{\mathbf{d} x \mathbf{d}}$ to map the spaces, where $\mathbf{d}$ is the word embedding dimension. Seeds in Procrustes solve Equation 1 to find an approximation W to W_{ideal}. Accordingly, the seeds can be thought of as samples from which one deduces the optimal linear transformation. This is a supervised learning problem, so when there are few seeds/samples, it is difficult to estimate W_{ideal}. Furthermore, the entire space X is mapped by W to a shared space with Y meaning that *every* point in X is subject to a potentially inaccurate mapping W: the mapping extrapolates to the entire space. As graph-based methods, GOAT and SGM do not suffer this issue and can induce non-linear relationships. Graph methods can be thought of as a semi-supervised learning problem: even words that don't serve as seeds are incorporated in the matching process. The graph manifold provides additional information that can be exploited.

Secondly, the dimension of the relationship between words in GOAT/SGM is much lower than for Procrustes. For the former, the relationship is one-dimensional: distance. As words for the Procrustes method are embedded in d-dimensional Euclidean space, their relationships have a magnitude *and* a direction: they are $\{d + 1\}$-dimensional. It is possible that the lower dimension in GOAT/SGM makes them robust to noise, explaining why the graph-based methods outperform Procrustes in low-resource settings. This hypothesis should be investigated in follow-up studies.

6 Related Work

BLI Recent years have seen a proliferation of the BLI literature (e.g. Ruder et al., 2018; Aldarmaki et al., 2018; Joulin et al., 2018; Doval et al., 2018; Artetxe et al., 2019a; Huang et al., 2019; Patra et al., 2019; Zhang et al., 2020; Biesialska and Ruiz Costa-Jussà, 2020). Many use Procrustes-based solutions, which assume that embedding spaces are roughly isomorphic. Wang et al. (2021) argue that the mapping can only be piece-wise linear, and induce multiple mappings. Ganesan et al. (2021) learn an "invertible neural network" as a non-linear mapping of spaces, and Cao and Zhao (2018) align spaces using point set registration. Many approaches address only high-resource languages. The tendency to evaluate on similar languages with high-quality data from similar domains hinders advancement in the field (Artetxe et al., 2020).

BLI with OT Most similar to ours are BLI approaches which incorporate OT formulations using the Sinkhorn and/or Hungarian algorithms (e.g. Alvarez-Melis and Jaakkola, 2018; Alaux et al., 2018). Grave et al. (2019) optimize "Procrustes in Wasserstein Distance", iteratively updating a linear transformation and permutation matrix using Frank-Wolfe on samples from embedding spaces $\mathbf{X}$ and $\mathbf{Y}$. Zhao et al. (2020b) and Zhang et al. (2017) also use an iterative procedure. Ramírez et al. (2020) combine Procrustes and their Iterative Hungarian algorithms. Xu et al. (2018) use Sinkhorn distance in the loss function, and (Zhang et al., 2017) use Sinkhorn to minimize distance between spaces. Haghighi et al. (2008) use the Hun-

	Prev	*-EP*	*-EG*	*Prev*	*-EP*	*-EG*	*Prev*	*-EP*	*-EG*	*Prev*	*-EP*	*-EG*	*Prev*	*-EP*	*-EG*	*Prev*	*-EP*	*-EG*
Seeds		en-bn			bn-en			en-bs			bs-en			en-de			de-en	
25	*31.9*	**44.3**	0.5	*44.4*	**53.1**	6.2	*48.1*	**58.4**	0.4	*54.6*	**61.5**	0.6	*48.5*	**61.7**	59.1	*34.1*	**59.3**	56.8
75	*33.1*	**44.7**	39.5	*45.2*	**53.5**	49.1	*47.9*	**58.1**	55.1	*54.6*	**61.7**	57.2	*48.8*	**62.3**	59.7	*47.0*	**59.4**	57.1
100	*33.8*	**45.3**	39.7	*45.5*	**53.9**	48.1	*47.7*	**58.1**	55.4	*55.5*	**61.3**	57.5	*49.0*	**62.2**	59.7	*47.6*	**59.4**	56.8
2000	*45.9*	48.7	**49.3**	*55.2*	**56.6**	56.3	*58.1*	59.9	**60.5**	*65.9*	66.6	**69.1**	*63.1*	66.3	**69.4**	*60.7*	65.1	**67.9**
4000	*60.3*	50.5	**61.2**	*65.9*	55.2	**68.9**	*70.6*	63.4	**71.8**	***86.1***	69.7	85.7	*74.2*	72.9	**79.5**	*71.4*	67.2	**77.6**
		en-et			et-en			en-fa			fa-en			en-id			id-en	
25	*47.0*	**60.4**	5.9	*63.2*	**70.2**	15.0	*42.7*	**54.4**	4.5	*45.1*	**55.1**	2.0	*51.3*	**65.8**	0.6	*58.0*	**66.7**	1.8
75	*47.5*	**60.6**	58.6	*64.2*	**69.8**	66.5	*42.9*	**54.1**	51.9	*45.8*	**55.3**	52.0	*53.6*	**66.0**	63.3	*57.0*	**66.7**	64.2
100	*47.3*	**61.1**	59.0	*64.3*	**70.2**	66.7	*43.0*	**54.3**	52.3	*45.6*	**55.5**	52.7	*54.3*	**66.2**	63.2	*57.8*	**67.1**	63.9
2000	*64.0*	66.6	**67.4**	*74.2*	74.1	**75.0**	*54.7*	58.0	**58.4**	*53.8*	**57.5**	56.9	*70.7*	72.1	**74.2**	*70.1*	72.5	**72.9**
4000	*77.4*	71.7	**80.2**	*86.4*	80.7	**87.2**	*65.9*	62.4	**67.4**	*65.5*	60.1	**67.0**	*84.3*	76.8	**86.2**	*80.5*	78.2	**83.7**
		en-mk			mk-en			en-ms			ms-en			en-ru			ru-en	
25	*55.8*	**63.9**	8.0	*63.2*	**68.8**	0.6	*36.6*	**62.6**	0.9	*38.6*	**65.3**	0.2	*55.7*	**67.7**	66.1	*54.4*	**63.9**	62.0
75	*56.5*	**64.3**	63.3	*63.6*	**68.8**	67.9	*42.6*	**62.6**	59.8	*56.8*	**65.6**	62.8	*55.7*	**68.1**	67.0	*54.8*	**63.9**	61.6
100	*56.2*	64.1	**64.2**	*64.4*	**69.4**	67.5	*42.9*	**63.0**	58.6	*57.7*	**65.8**	63.1	*55.9*	**67.9**	66.4	*55.1*	**63.8**	61.3
2000	*64.6*	66.8	**67.9**	*71.7*	71.0	**73.1**	*65.0*	67.0	**68.5**	*67.4*	69.4	**69.7**	*67.5*	72.6	**74.2**	*68.5*	69.3	**72.5**
4000	*75.6*	68.9	**77.1**	*88.8*	74.1	**91.1**	*79.3*	70.7	**79.5**	*77.4*	70.0	**79.1**	*83.3*	79.3	**86.5**	***89.3***	77.4	**89.3**
		en-ta			ta-en			en-vi			vi-en			en-zh			zh-en	
25	*1.8*	**2.2**	0.6	*44.4*	**51.4**	2.4	*0.4*	0.4	0.2	*3.3*	**5.3**	0.2	*8.7*	**52.7**	1.7	*9.3*	**48.1**	0.8
75	*30.5*	**40.4**	35.7	*45.1*	**52.4**	46.9	*22.9*	**55.0**	1.2	*45.2*	**59.6**	54.4	*17.4*	**51.6**	46.4	*14.1*	**51.1**	48.0
100	*30.6*	**40.2**	36.8	*45.4*	**52.5**	47.9	*30.2*	**55.6**	36.2	*47.1*	**59.3**	56.3	*18.5*	**51.6**	47.3	*15.2*	**51.0**	48.0
2000	*40.6*	42.7	**44.2**	*55.1*	55.3	**56.2**	*61.1*	**67.3**	65.8	*66.3*	**73.5**	71.5	*49.3*	**58.0**	57.7	*41.8*	**57.6**	56.8
4000	*49.1*	44.0	**51.7**	***73.8***	65.3	71.6	*77.9*	73.0	**80.5**	*82.1*	80.1	**84.3**	*67.5*	66.4	**75.1**	*66.2*	65.3	**73.3**

Table 3: P@1 of Combination Exps. -EP starts with GOAT, ends with IterProc. -EG: IterProc, ends with GOAT. *Prev* is previous best of prior experiments. Some seed sizes omitted for brevity (see Appendix).

garian Algorithm for BLI from text. Lian et al. and Alaux et al. (2018) align all languages to a common space for multilingual BLI. The latter use Sinkhorn to approximate a permutation matrix in their formulation. Zhao et al. (2020a) incorporate OT for semi-supervised BLI.

7 Conclusion

We perform bilingual lexicon induction from word embedding spaces of 40 language pairs, utilizing the newly-developed GOAT algorithm for graph-matching. Performance is strong across all pairs, especially on dissimilar languages with low-supervision. As the major use case of low-resource BLI and MT is dissimilar languages with low supervision, the strong performance of GOAT is an encouraging result for real-world applications.

Limitations

Although we evaluate GOAT on 40 language pairs, this does not capture the full linguistic diversity of world languages. Languages of Eurasia are overrepresented, particularly the Indo-European family. Each pair has at least one high-resource Indo-European language which uses the Latin script. Future work should examine GOAT's performance when both languages are low-resource, and on an even broader diversity of languages from around the world. Furthermore, graph-matching methods are also considerably slower than calculating the solution to the orthogonal Procustes problem on GPU, potentially limiting the former's usefulness when one must match large sets of words. Future work might examine the speed/accuracy trade-off between methods as embedding space size scales.

Acknowledgements

This material is based on research sponsored by the Air Force Research Laboratory and Defense Advanced Research Projects Agency (DARPA) under agreement number FA8750-20-2-1001. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory and DARPA or the U.S. Government. This material is based upon work supported by the United States Air Force under Contract No. FA8750-19-C-0098. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force and DARPA.

References

Jean Alaux, Edouard Grave, Marco Cuturi, and Armand Joulin. 2018. Unsupervised hyper-alignment for multilingual word embeddings. In *International Conference on Learning Representations*.

Hanan Aldarmaki, Mahesh Mohan, and Mona Diab. 2018. Unsupervised word mapping using structural similarities in monolingual embeddings. *Transactions of the Association for Computational Linguistics*, 6:185–196.

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

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 789–798, Melbourne, Australia. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2019a. Bilingual lexicon induction through unsupervised machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5002–5007, Florence, Italy. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2019b. An effective approach to unsupervised machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 194–203, Florence, Italy. Association for Computational Linguistics.

Mikel Artetxe, Sebastian Ruder, Dani Yogatama, Gorka Labaka, and Eneko Agirre. 2020. A call for more rigor in unsupervised cross-lingual learning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7375–7388, Online. Association for Computational Linguistics.

Magdalena Marta Biesialska and Marta Ruiz Costa-Jussà. 2020. Refinement of unsupervised cross-lingual word embeddings. In *ECAI 2020, 24th European Conference on Artificial Intelligence: 29 August–8 September 2020, Santiago de Compostela, Spain: including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020): proceedings*, pages 1–4. Ios Press.

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

Hailong Cao and Tiejun Zhao. 2018. Point set registration for unsupervised bilingual lexicon induction. In *IJCAI*, pages 3991–3997.

Jaewon Chung, Benjamin D Pedigo, Eric W Bridgeford, Bijan K Varjavand, Hayden S Helm, and Joshua T Vogelstein. 2019. Graspy: Graph statistics in python. *Journal of Machine Learning Research*, 20(158):1–7.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Marco Cuturi. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26:2292–2300.

Paula Czarnowska, Sebastian Ruder, Édouard Grave, Ryan Cotterell, and Ann Copestake. 2019. Don't forget the long tail! a comprehensive analysis of morphological generalization in bilingual lexicon induction. 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)*, pages 974–983.

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

Donniell E Fishkind, Sancar Adali, Heather G Patsolic, Lingyao Meng, Digvijay Singh, Vince Lyzinski, and Carey E Priebe. 2019. Seeded graph matching. *Pattern recognition*, 87:203–215.

Marguerite Frank, Philip Wolfe, et al. 1956. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110.

Ashwinkumar Ganesan, Francis Ferraro, and Tim Oates. 2021. Learning a reversible embedding mapping using bi-directional manifold alignment. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3132–3139, Online. Association for Computational Linguistics.

Goran Glavaš, Robert Litschko, Sebastian Ruder, and Ivan Vulić. 2019. 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*, pages 710–721, Florence, Italy. Association for Computational Linguistics.

Chengyue Gong, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2018. Frage: Frequency-agnostic word representation. *arXiv preprint arXiv:1809.06858*.

John C Gower, Garmt B Dijksterhuis, et al. 2004. *Procrustes problems*, volume 30. Oxford University Press on Demand.

Edouard Grave, Armand Joulin, and Quentin Berthet. 2019. Unsupervised alignment of embeddings with wasserstein procrustes. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1880–1890. PMLR.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proceedings of ACL-08: HLT*, pages 771–779, Columbus, Ohio. Association for Computational Linguistics.

Jiaji Huang, Qiang Qiu, and Kenneth Church. 2019. Hubless nearest neighbor search for bilingual lexicon induction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4072–4080, Florence, Italy. Association for Computational Linguistics.

Roy Jonker and Anton Volgenant. 1987. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340.

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

Harold W Kuhn. 1955. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

Xin Lian, Kshitij Jain, Jakub Truszkowski, Pascal Poupart, and Yaoliang Yu. Unsupervised multilingual alignment using wasserstein barycenter.

Kelly Marchisio, Kevin Duh, and Philipp Koehn. 2020. When does unsupervised machine translation work? In *Proceedings of the Fifth Conference on Machine Translation*, pages 571–583, Online. Association for Computational Linguistics.

Kelly Marchisio, Youngser Park, Ali Saad-Eldin, Anton Alyakin, Kevin Duh, Carey Priebe, and Philipp Koehn. 2021. An analysis of Euclidean vs. graph-based framing for bilingual lexicon induction from word embedding spaces. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 738–749, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

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

Aitor Ormazabal, Mikel Artetxe, Gorka Labaka, Aitor Soroa, and Eneko Agirre. 2019. Analyzing the limitations of cross-lingual word embedding mappings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4990–4995, Florence, Italy. Association for Computational Linguistics.

Barun Patra, Joel Ruben Antony Moniz, Sarthak Garg, Matthew R. Gormley, and Graham Neubig. 2019. Bilingual lexicon induction with semi-supervision in non-isometric embedding spaces. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 184–193, Florence, Italy. Association for Computational Linguistics.

Gabriel Peyre and Marco Cuturi. 2019. Computational optimal transport. *Foundations and Trends in Machine Learning*, 11(5-6):355–607.

Guillem Ramírez, Rumen Dangovski, Preslav Nakov, and Marin Soljačić. 2020. On a novel application of wasserstein-procrustes for unsupervised cross-lingual learning. *arXiv preprint arXiv:2007.09456*.

Sebastian Ruder, Ryan Cotterell, Yova Kementchedjhieva, and Anders Søgaard. 2018. A discriminative latent-variable model for bilingual lexicon induction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 458–468, Brussels, Belgium. Association for Computational Linguistics.

Ali Saad-Eldin, Benjamin D Pedigo, Carey E Priebe, and Joshua T Vogelstein. 2021. Graph matching via optimal transport. *arXiv preprint arXiv:2111.05366*.

Peter H Schönemann. 1966. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10.

Haoyue Shi, Luke Zettlemoyer, and Sida I. Wang. 2021. Bilingual lexicon induction via unsupervised bitext construction and word alignment. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 813–826, Online. Association for Computational Linguistics.

Richard Sinkhorn. 1967. Diagonal equivalence to matrices with prescribed row and column sums. *The American Mathematical Monthly*, 74(4):402–405.

Anders Søgaard, Sebastian Ruder, and Ivan Vulić. 2018. On the limitations of unsupervised bilingual dictionary induction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 778–788, Melbourne, Australia. Association for Computational Linguistics.

Joshua T Vogelstein, John M Conroy, Vince Lyzinski, Louis J Podrazik, Steven G Kratzer, Eric T Harley, Donniell E Fishkind, R Jacob Vogelstein, and Carey E Priebe. 2015. Fast approximate quadratic programming for graph matching. *PLOS one*, 10(4):e0121002.

Ivan Vulić, Goran Glavaš, Roi Reichart, and Anna Korhonen. 2019. Do we really need fully unsupervised cross-lingual embeddings? 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)*, pages 4407–4418, Hong Kong, China. Association for Computational Linguistics.

Ivan Vulić, Sebastian Ruder, and Anders Søgaard. 2020. Are all good word vector spaces isomorphic? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3178–3192, Online. Association for Computational Linguistics.

Haozhou Wang, James Henderson, and Paola Merlo. 2021. Multi-adversarial learning for cross-lingual word embeddings. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 463–472, Online. Association for Computational Linguistics.

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

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

Mozhi Zhang, Yoshinari Fujinuma, Michael J. Paul, and Jordan Boyd-Graber. 2020. Why overfitting isn't always bad: Retrofitting cross-lingual word embeddings to dictionaries. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2214–2220, Online. Association for Computational Linguistics.

Mozhi Zhang, Keyulu Xu, Ken-ichi Kawarabayashi, Stefanie Jegelka, and Jordan Boyd-Graber. 2019. Are girls neko or shōjo? cross-lingual alignment of non-isomorphic embeddings with iterative normalization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3180–3189, Florence, Italy. Association for Computational Linguistics.

Xu Zhao, Zihao Wang, Hao Wu, and Yong Zhang. 2020a. Semi-supervised bilingual lexicon induction with two-way interaction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2973–2984, Online. Association for Computational Linguistics.

Xu Zhao, Zihao Wang, Yong Zhang, and Hao Wu. 2020b. A relaxed matching procedure for unsupervised BLI. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3036–3041, Online. Association for Computational Linguistics.

Appendix

	*-to-en		en-to-*		
	Full	1-1	Full	1-1	# Embs
bn	7588	4299	8467	4556	145350
bs	6164	4294	8153	4795	166505
de	10866	4451	14677	4903	200000
en	n/a	n/a	n/a	n/a	200000
es	8667	4445	11977	4866	200000
et	6509	4352	8261	4738	200000
fa	8510	4582	8869	4595	200000
fr	8270	4548	10872	4827	200000
id	9677	4563	9407	4573	200000
it	7364	4478	9657	4815	200000
ja	6819	4112	7135	4351	200000
mk	7197	4259	10075	4820	176947
ms	8140	4650	7394	4454	155629
ru	7452	4084	10887	4812	200000
ta	6850	4225	8091	4744	200000
vi	7251	4775	6353	4507	200000
zh	8891	4450	8728	4381	200000

Table A1: Size of train/test sets before (Full) & after making one-to-one (1-1), with # of embeddings used.

	en-de		ru-en	
Seeds	Rand.	Bary.	Rand.	Bary.
100	45.7	45.9	49.6	50.4
200	47.4	47.5	52.5	52.5
500	52.3	51.9	55.4	55.6
1000	54.6	54.9	58.3	58.1
2000	61.5	61.4	67.1	67.1
4000	74.2	74.2	89.3	89.3

Table A2: SGM with barycenter vs. randomized initialization for languages used in Marchisio et al. (2021). The difference is negligible.

Unsupervised Performance For some highly-related languages, GOAT performs well even with no seeds (unsupervised). GOAT scores 48.8 on English→German, 34.5 on German→English, 62.4 on English→Spanish, and 19.6 on Spanish→English with no supervision. Particularly striking is the unsupervised performance on highly-related languages: >87 on Italian↔French and >90 for Spanish↔Portuguese. We suspect that that the word embedding spaces are highly isomorphic for these language pairs, allowing GOAT (and sometimes SGM) to easily recover the translations.

Iterative Results of Iterative Procrustes (IterProc), Iterative SGM (IterSGM), and Iterative GOAT (IterGOAT) are in Table A5. We run the Iterative Procrustes and Iterative SGM procedures of Marchisio et al. (2021) with stochastic-add. Here, Procrustes [or SGM] is run in source↔target directions, hypotheses are intersected, and H random hypotheses are added to the gold seeds and fed into subsequent runs of Procrustes [SGM]. The next iteration adds 2H hypotheses, repeating until all hypotheses are chosen. We set $H = 100$ and create an analogous iterative algorithm for GOAT, which we call Iterative GOAT.

IterSGM/GOAT perform similarly across conditions, with a few exceptions where either performs very strongly with no supervision: IterGOAT scores 49.2, 45.2, 34.4, 58.2, and 55.9 for En-De, En-Fa, De-En, Id-En, and Ru-En, respectively, and IterSGM scores 57.3 for En-Ru. On Chinese↔English, IterGOAT underperforms IterSGM, similar to GOAT's underperformance of SGM in the single run.

Similar to Marchisio et al. (2021), we find that IterProc compensates for an initial poor first run and outperforms IterSGM with a moderate amount of seeds (100+). Extending to the very lowest seeds sizes (0-75), however, IterSGM/IterGOAT are superior. With 25 seeds, IterProc fails for all language pairs except En↔De and En↔Ru, scoring $P@1 < 5$. IterSGM and IterGOAT, however, perform reasonably well for most language pairs with 25 seeds, suggesting that the graph-based framing is the better approach for low-seed levels. At the highest supervision level (2000+ seeds), IterSGM/IterGOAT again tends to be superior.

Differences are minor btwn using GOAT or SGM in the iterative or system combination experiments. This results suggests that mixing Procrustes and graph-based framings is helpful for BLI, regardless of which algorithm one picks. It is interesting to contemplate what other problems might benefit from examination from multiple mathematical framings in one solution, as each may have complementary benefits.

	en-bn				en-bs				en-et				en-fa				en-id				en-mk				en-ms				en-ta			
Seeds	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ
0	0.0	0.0	**0.1**	*(+0.1)*	0.0	0.0	0.0	*(+0.0)*	0.0	0.0	**0.2**	*(+0.2)*	0.0	0.2	**0.5**	*(+0.3)*	0.0	0.2	**2.9**	*(+2.7)*	0.1	0.0	**0.1**	*(+0.1)*	0.0	0.0	**0.1**	*(+0.1)*	0.0	0.0	0.0	*(+0.0)*
25	0.0	12.9	**31.9**	*(+19.0)*	0.1	37.0	**48.1**	*(+11.1)*	0.1	37.5	**47.0**	*(+9.5)*	0.2	35.2	**42.7**	*(+7.5)*	0.1	49.2	**51.3**	*(+2.1)*	0.1	51.0	**55.8**	*(+4.8)*	0.3	24.9	**36.6**	*(+11.7)*	0.1	1.5	**1.8**	*(+0.3)*
50	0.2	24.7	**33.2**	*(+8.5)*	0.3	39.6	**47.9**	*(+8.3)*	0.7	39.1	**47.4**	*(+8.3)*	0.6	37.0	**42.6**	*(+5.6)*	0.4	46.8	**52.9**	*(+6.1)*	0.4	51.2	**56.2**	*(+5.0)*	0.5	29.3	**38.9**	*(+9.6)*	0.2	21.5	**30.5**	*(+9.0)*
75	0.5	25.2	**33.1**	*(+7.9)*	0.6	40.4	**47.9**	*(+7.5)*	1.3	40.6	**47.5**	*(+6.9)*	0.9	37.1	**42.9**	*(+5.8)*	1.1	49.8	**53.6**	*(+3.8)*	1.0	51.0	**56.5**	*(+5.5)*	1.2	37.0	**42.6**	*(+5.6)*	0.7	22.8	**30.5**	*(+7.7)*
100	0.7	27.8	**33.8**	*(+6.0)*	0.7	40.7	**47.7**	*(+7.0)*	1.7	41.2	**47.3**	*(+6.1)*	1.6	36.7	**43.0**	*(+6.3)*	2.3	51.1	**54.3**	*(+3.2)*	1.6	52.0	**56.2**	*(+4.2)*	1.9	39.1	**42.9**	*(+3.8)*	1.0	23.6	**30.6**	*(+7.0)*
200	4.4	30.9	**35.4**	*(+4.5)*	7.0	44.5	**49.9**	*(+5.4)*	9.0	46.0	**49.6**	*(+3.6)*	7.0	40.0	**44.4**	*(+4.4)*	13.0	54.1	**55.9**	*(+1.8)*	12.2	53.3	**57.6**	*(+4.3)*	11.5	43.3	**46.0**	*(+2.7)*	4.1	26.7	**31.0**	*(+4.3)*
500	20.3	37.9	**39.7**	*(+1.8)*	26.1	50.6	**52.7**	*(+2.1)*	31.1	53.2	**54.4**	*(+1.2)*	26.8	44.7	**46.8**	*(+2.1)*	43.8	**60.2**	59.2	*(-1.0)*	39.0	58.4	**60.3**	*(+1.9)*	37.1	**52.6**	51.5	*(-1.1)*	17.0	33.5	**33.9**	*(+0.4)*
1000	30.6	40.7	**41.3**	*(+0.6)*	39.1	54.4	**55.1**	*(+0.7)*	45.5	57.0	**57.3**	*(+0.3)*	40.9	48.2	**49.9**	*(+1.7)*	58.0	**64.5**	63.6	*(-0.9)*	51.0	60.3	**61.8**	*(+1.5)*	48.9	**58.9**	58.3	*(-0.6)*	26.9	**36.2**	**36.2**	*(+0.0)*
2000	36.9	**45.9**	45.5	*(-0.4)*	45.9	**58.1**	57.1	*(-1.0)*	53.1	63.4	**64.0**	*(+0.6)*	47.6	54.1	**54.7**	*(+0.6)*	65.0	**70.7**	69.5	*(-1.2)*	56.3	63.9	**64.6**	*(+0.7)*	55.3	**65.0**	63.0	*(-2.0)*	32.3	**40.6**	39.8	*(-0.8)*
4000	38.3	**60.3**	59.0	*(-1.3)*	48.4	**70.6**	69.8	*(-0.8)*	59.3	77.1	**77.4**	*(+0.3)*	51.1	65.5	**65.9**	*(+0.4)*	71.4	84.1	**84.3**	*(+0.2)*	59.9	**75.6**	75.5	*(-0.1)*	58.6	**79.3**	79.1	*(-0.2)*	33.5	**49.1**	48.5	*(-0.6)*

	en-vi				en-zh				en-ru				en-de				en-fr				en-es				en-it				en-ja			
Seeds	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ
0	0.1	0.1	0.0	*(-0.1)*	0.0	0.0	0.0	*(+0.0)*	0.0	0.0	0.0	*(+0.0)*	0.0	0.4	**48.8**	*(+48.4)*	0.0	**0.5**	0.4	*(-0.1)*	0.0	0.2	**62.4**	*(+62.2)*	0.0	0.1	**1.3**	*(+1.2)*	0.0	0.0	0.0	*(+0.0)*
25	0.2	**0.4**	**0.4**	*(+0.0)*	0.3	**8.7**	7.9	*(-0.8)*	0.4	49.6	**55.7**	*(+6.1)*	0.3	44.8	**48.5**	*(+3.7)*	0.4	58.6	**62.4**	*(+3.8)*	0.3	59.5	**62.8**	*(+3.3)*	0.4	60.0	**63.0**	*(+3.0)*	0.0	**1.1**	1.0	*(-0.1)*
50	0.1	1.6	**4.0**	*(+2.4)*	0.6	**15.6**	15.0	*(-0.6)*	1.0	50.2	**55.6**	*(+5.4)*	0.8	44.6	**48.7**	*(+4.1)*	1.1	58.1	**62.6**	*(+4.5)*	1.1	58.3	**62.4**	*(+4.1)*	1.4	58.3	**62.4**	*(+4.1)*	0.4	9.1	**9.3**	*(+0.2)*
75	0.3	13.8	**22.9**	*(+9.1)*	1.6	**17.4**	14.8	*(-2.6)*	1.9	50.5	**55.7**	*(+5.2)*	2.6	45.3	**48.8**	*(+3.5)*	3.0	59.2	**62.5**	*(+3.3)*	3.0	60.2	**63.6**	*(+3.4)*	3.4	61.7	**63.7**	*(+2.0)*	0.9	10.3	**10.7**	*(+0.4)*
100	0.5	18.3	**30.2**	*(+11.9)*	3.1	**18.5**	15.3	*(-3.2)*	3.1	51.7	**55.9**	*(+4.2)*	3.6	45.9	**49.0**	*(+3.1)*	5.8	59.4	**62.3**	*(+2.9)*	5.0	60.5	**63.3**	*(+2.8)*	5.9	60.7	**63.3**	*(+2.6)*	1.3	**14.4**	13.0	*(-1.4)*
200	2.9	34.8	**40.6**	*(+5.8)*	12.0	**22.7**	17.3	*(-5.4)*	16.5	54.3	**57.3**	*(+3.0)*	16.1	47.5	**50.2**	*(+2.7)*	24.9	60.9	**63.2**	*(+2.3)*	27.2	62.1	**63.9**	*(+1.8)*	24.9	63.2	**64.4**	*(+1.2)*	8.3	**20.7**	15.6	*(-5.1)*
500	19.2	43.4	**47.9**	*(+4.5)*	**33.9**	30.3	22.5	*(-7.8)*	45.8	58.7	**59.4**	*(+0.7)*	44.9	51.9	**52.4**	*(+0.5)*	57.9	65.2	**65.4**	*(+0.2)*	59.4	65.7	**66.1**	*(+0.4)*	57.3	**67.9**	67.5	*(-0.4)*	**33.3**	28.3	20.5	*(-7.8)*
1000	37.4	53.7	**54.9**	*(+1.2)*	**44.4**	34.5	29.5	*(-5.0)*	58.3	**61.7**	61.5	*(-0.2)*	**57.2**	54.9	54.5	*(-0.4)*	67.9	**68.3**	67.9	*(-0.4)*	**69.6**	68.8	69.0	*(+0.2)*	69.8	**70.5**	70.3	*(-0.2)*	**47.8**	35.6	27.4	*(-8.2)*
2000	50.7	60.7	**61.1**	*(+0.4)*	**49.3**	45.6	41.7	*(-3.9)*	65.8	67.4	**67.5**	*(+0.1)*	**63.1**	61.4	61.8	*(+0.4)*	72.5	**72.9**	72.2	*(-0.7)*	74.1	**75.2**	74.9	*(-0.3)*	75.6	75.9	**76.0**	*(+0.1)*	**56.0**	45.1	40.3	*(-4.8)*
4000	59.4	**77.9**	77.7	*(-0.2)*	58.5	**67.5**	**67.5**	*(+0.0)*	73.4	**83.3**	81.8	*(-1.5)*	70.8	**74.2**	74.1	*(-0.1)*	78.2	**82.3**	**82.3**	*(+0.0)*	78.6	**83.1**	82.7	*(-0.4)*	77.5	**84.3**	83.9	*(-0.4)*	61.5	68.7	**68.9**	*(+0.2)*

	bn-en				bs-en				et-en				fa-en				id-en				mk-en				ms-en				ta-en			
Seeds	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ
0	0.0	0.0	0.0	*(+0.0)*	0.0	0.0	0.0	*(+0.0)*	0.0	0.1	0.0	*(-0.1)*	0.0	0.0	0.1	*(+0.1)*	0.0	0.1	**1.0**	*(+0.9)*	0.0	0.0	0.0	*(+0.0)*	0.0	0.0	0.0	*(+0.0)*	0.0	0.0	**0.5**	*(+0.5)*
25	0.2	37.2	**44.4**	*(+7.2)*	0.2	44.7	**54.6**	*(+9.9)*	0.3	55.9	**63.2**	*(+7.3)*	0.1	37.1	**45.1**	*(+8.0)*	0.1	54.6	**58.0**	*(+3.4)*	0.2	60.1	**63.2**	*(+3.1)*	0.2	10.0	**38.6**	*(+28.6)*	0.2	35.2	**44.4**	*(+9.2)*
50	0.5	39.2	**45.0**	*(+5.8)*	0.3	45.2	**54.6**	*(+9.4)*	0.8	56.0	**63.3**	*(+7.3)*	0.3	38.2	**45.3**	*(+7.1)*	0.5	52.1	**57.1**	*(+5.0)*	1.2	60.1	**63.5**	*(+3.4)*	0.6	51.1	**57.2**	*(+6.1)*	0.6	37.7	**45.1**	*(+7.4)*
75	1.0	39.9	**45.2**	*(+5.3)*	1.1	47.7	**54.6**	*(+6.9)*	1.7	57.9	**64.2**	*(+6.3)*	0.8	39.7	**45.8**	*(+6.1)*	1.4	52.3	**57.0**	*(+4.7)*	1.8	60.5	**63.6**	*(+3.1)*	1.9	51.9	**56.8**	*(+4.9)*	1.1	40.2	**45.1**	*(+4.9)*
100	1.4	40.3	**45.5**	*(+5.2)*	1.9	48.5	**55.5**	*(+7.0)*	3.2	58.3	**64.3**	*(+6.0)*	1.5	39.8	**45.6**	*(+5.8)*	2.9	53.2	**57.8**	*(+4.6)*	4.4	61.0	**64.4**	*(+3.4)*	2.9	53.7	**57.7**	*(+4.0)*	2.1	39.9	**45.4**	*(+5.5)*
200	7.6	43.0	**46.6**	*(+3.6)*	6.8	50.4	**56.2**	*(+5.8)*	12.6	60.7	**64.8**	*(+4.1)*	9.9	42.7	**46.8**	*(+4.1)*	13.3	55.7	**58.3**	*(+2.6)*	16.5	62.0	**64.5**	*(+2.5)*	15.5	56.2	**58.9**	*(+2.7)*	7.4	43.4	**45.8**	*(+2.4)*
500	26.7	48.1	**48.7**	*(+0.6)*	28.0	55.4	**58.9**	*(+3.5)*	40.0	64.4	**67.1**	*(+2.7)*	33.4	46.6	**48.0**	*(+1.4)*	47.3	60.1	**61.1**	*(+1.0)*	48.6	65.2	**66.3**	*(+1.1)*	43.7	58.6	**60.7**	*(+2.1)*	25.5	47.0	**48.2**	*(+1.2)*
1000	39.0	**51.5**	51.3	*(-0.2)*	44.7	59.2	**61.6**	*(+2.4)*	54.6	67.7	**70.0**	*(+2.3)*	45.9	48.5	**49.6**	*(+1.1)*	58.5	63.2	**64.1**	*(+0.9)*	58.5	66.1	**67.9**	*(+1.8)*	55.7	61.6	**62.3**	*(+0.7)*	37.1	49.6	**50.3**	*(+0.7)*
2000	45.2	**55.2**	54.2	*(-1.0)*	54.5	64.8	**65.9**	*(+1.1)*	62.6	72.8	**74.2**	*(+1.4)*	50.3	**53.8**	53.6	*(-0.2)*	66.0	**70.1**	**70.1**	*(+0.0)*	62.6	**71.7**	71.6	*(-0.1)*	60.2	**67.4**	67.2	*(-0.2)*	45.2	**55.1**	53.5	*(-1.6)*
4000	44.8	**65.9**	64.9	*(-1.0)*	54.4	83.7	**86.1**	*(+2.4)*	67.0	**86.4**	86.1	*(-0.3)*	52.1	**65.5**	64.8	*(-0.7)*	73.2	**80.5**	**80.5**	*(+0.0)*	65.3	88.4	**88.8**	*(+0.4)*	63.1	**77.4**	77.1	*(-0.3)*	48.4	**73.8**	73.3	*(-0.5)*

Table A3: Full Results (1 of 2): P@1 of Procrustes (P), SGM (S) or GOAT (G). Δ is gain/loss of GOAT over SGM.

	vi-en				zh-en				ru-en				de-en			
Seeds	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ
0	0.0	0.0	0.1	*(+0.1)*	0.0	0.0	0.0	*(+0.0)*	0.0	0.4	**2.8**	*(+2.4)*	0.0	0.6	**34.5**	*(+33.9)*
25	0.1	1.0	**3.3**	*(+2.3)*	0.1	6.8	**9.3**	*(+2.5)*	0.3	49.1	**54.4**	*(+5.3)*	0.2	30.3	**34.1**	*(+3.8)*
50	0.2	27.4	**41.7**	*(+14.3)*	0.6	**13.1**	12.5	*(-0.6)*	1.0	49.5	**54.8**	*(+5.3)*	0.6	42.5	**47.6**	*(+5.1)*
75	0.3	29.8	**45.2**	*(+15.4)*	1.3	12.6	**14.1**	*(+1.5)*	2.1	49.9	**54.8**	*(+4.9)*	1.5	42.6	**47.0**	*(+4.4)*
100	0.5	34.4	**47.1**	*(+12.7)*	2.4	**15.2**	14.9	*(-0.3)*	4.1	50.4	**55.1**	*(+4.7)*	3.0	41.9	**47.6**	*(+5.7)*
200	1.7	41.3	**48.6**	*(+7.3)*	12.0	**19.8**	16.1	*(-3.7)*	16.6	52.5	**55.2**	*(+2.7)*	10.6	45.0	**48.3**	*(+3.3)*
500	8.5	45.7	**51.7**	*(+6.0)*	**33.0**	27.9	25.0	*(-2.9)*	45.3	55.6	**57.3**	*(+1.7)*	39.5	49.1	**50.3**	*(+1.2)*
1000	29.1	52.4	**57.2**	*(+4.8)*	**44.5**	33.2	31.9	*(-1.3)*	56.6	58.1	**60.3**	*(+2.2)*	52.8	53.1	**53.3**	*(+0.2)*
2000	56.3	64.1	**66.3**	*(+2.2)*	**50.8**	41.5	41.8	*(+0.3)*	62.7	67.1	**68.5**	*(+1.4)*	**60.7**	60.5	60.4	*(-0.1)*
4000	70.3	81.9	**82.1**	*(+0.2)*	58.4	**66.2**	**66.2**	*(+0.0)*	67.9	**89.3**	**89.3**	*(+0.0)*	63.6	**71.4**	71.0	*(-0.4)*
	fr-en				es-en				it-en				ja-en			
Seeds	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ
0	0.0	0.2	**1.1**	*(+0.9)*	0.0	0.6	**19.6**	*(+19.0)*	0.0	0.1	**0.4**	*(+0.3)*	0.1	0.0	0.0	*(+0.0)*
25	0.3	62.5	**65.3**	*(+2.8)*	0.4	61.4	**64.4**	*(+3.0)*	0.4	63.6	**66.9**	*(+3.3)*	0.2	8.0	**31.2**	*(+23.2)*
50	1.4	62.8	**65.5**	*(+2.7)*	2.4	61.9	**65.0**	*(+3.1)*	1.3	63.5	**67.0**	*(+3.5)*	1.6	22.0	**30.5**	*(+8.5)*
75	4.6	62.5	**65.7**	*(+3.2)*	4.7	62.8	**64.9**	*(+2.1)*	3.1	64.2	**67.5**	*(+3.3)*	3.0	24.1	**31.1**	*(+7.0)*
100	6.8	63.0	**65.9**	*(+2.9)*	7.8	63.5	**65.0**	*(+1.5)*	7.2	64.9	**67.9**	*(+3.0)*	5.7	27.9	**31.9**	*(+4.0)*
200	26.8	64.2	**66.4**	*(+2.2)*	32.6	65.0	**65.9**	*(+0.9)*	28.2	66.6	**68.7**	*(+2.1)*	19.1	**34.6**	34.2	*(-0.4)*
500	62.4	67.7	**68.5**	*(+0.8)*	61.5	67.1	**68.2**	*(+1.1)*	61.9	69.8	**70.4**	*(+0.6)*	38.3	**39.9**	37.2	*(-2.7)*
1000	**71.4**	70.1	70.4	*(+0.3)*	69.5	69.8	**70.6**	*(+0.8)*	71.6	71.8	**73.0**	*(+1.2)*	**47.7**	42.7	38.9	*(-3.8)*
2000	**76.2**	75.3	74.8	*(-0.5)*	72.9	**74.5**	73.9	*(-0.6)*	**76.3**	75.6	75.8	*(+0.2)*	**54.0**	48.6	47.6	*(-1.0)*
4000	82.7	**90.0**	89.6	*(-0.4)*	74.4	**86.3**	85.8	*(-0.5)*	81.8	86.8	**87.2**	*(+0.4)*	54.5	**72.3**	70.5	*(-1.8)*
	de-es				it-fr				es-pt				pt-de			
Seeds	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ
0	0.0	0.1	0.8	*(+0.7)*	0.0	0.1	**87.6**	*(+87.5)*	0.0	94.5	**94.9**	*(+0.4)*	0.0	0.0	0.0	*(+0.0)*
25	0.5	67.4	**70.5**	*(+3.1)*	1.1	85.4	**87.3**	*(+1.9)*	2.5	94.3	**94.9**	*(+0.6)*	0.5	72.5	**76.2**	*(+3.7)*
50	1.6	67.6	**70.0**	*(+2.4)*	4.4	85.1	**87.3**	*(+2.2)*	10.4	94.4	**95.0**	*(+0.6)*	1.6	72.5	**75.8**	*(+3.3)*
75	4.2	65.7	**69.5**	*(+3.8)*	12.9	85.3	**87.2**	*(+1.9)*	23.7	94.4	**94.9**	*(+0.5)*	3.4	72.8	**76.0**	*(+3.2)*
100	7.0	65.8	**69.8**	*(+4.0)*	21.6	85.8	**87.4**	*(+1.6)*	38.6	94.8	**95.1**	*(+0.3)*	6.4	72.9	**76.4**	*(+3.5)*
200	26.0	67.9	**71.0**	*(+3.1)*	57.3	86.7	**87.7**	*(+1.0)*	74.1	94.9	**95.2**	*(+0.3)*	36.3	74.8	**76.9**	*(+2.1)*
500	59.9	72.6	**73.5**	*(+0.9)*	81.5	87.8	**88.4**	*(+0.6)*	90.6	**95.4**	**95.4**	*(+0.0)*	68.0	77.2	**77.7**	*(+0.5)*
1000	71.1	74.0	**75.4**	*(+1.4)*	86.1	89.1	**89.4**	*(+0.3)*	93.3	95.8	**96.0**	*(+0.2)*	75.5	78.9	**79.6**	*(+0.7)*
2000	76.1	78.8	**79.1**	*(+0.3)*	88.8	89.6	**90.4**	*(+0.8)*	94.5	97.0	**97.1**	*(+0.1)*	80.0	81.9	**82.0**	*(+0.1)*
4000	79.3	88.6	**89.1**	*(+0.5)*	90.0	92.8	**92.9**	*(+0.1)*	96.3	**98.9**	**98.9**	*(+0.0)*	82.8	**88.6**	87.9	*(-0.7)*
	es-de				fr-it				pt-es				de-pt			
Seeds	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ	*P*	*S*	*G*	Δ
0	0.0	0.0	0.0	*(+0.0)*	0.0	89.0	**90.3**	*(+1.3)*	0.1	1.1	**90.4**	*(+89.3)*	0.0	0.1	0.2	*(+0.1)*
25	0.5	62.1	**66.2**	*(+4.1)*	0.6	88.9	**90.2**	*(+1.3)*	1.1	89.7	**90.5**	*(+0.8)*	0.3	72.9	**76.2**	*(+3.3)*
50	1.3	62.4	**66.1**	*(+3.7)*	3.6	89.2	**90.6**	*(+1.4)*	8.3	89.7	**90.7**	*(+1.0)*	1.4	72.5	**76.3**	*(+3.8)*
75	4.2	63.0	**66.2**	*(+3.2)*	10.2	89.6	**91.0**	*(+1.4)*	15.7	89.4	**90.7**	*(+1.3)*	3.4	71.6	**76.4**	*(+4.8)*
100	7.8	63.6	**66.2**	*(+2.6)*	20.9	89.8	**91.0**	*(+1.2)*	25.4	89.7	**90.8**	*(+1.1)*	4.8	71.7	**76.3**	*(+4.6)*
200	27.7	65.2	**67.2**	*(+2.0)*	58.4	90.4	**91.2**	*(+0.8)*	70.0	90.4	**90.7**	*(+0.3)*	24.3	73.9	**77.1**	*(+3.2)*
500	59.9	67.5	**68.4**	*(+0.9)*	83.9	91.6	**91.8**	*(+0.2)*	86.6	90.2	**90.8**	*(+0.6)*	62.4	76.9	**78.6**	*(+1.7)*
1000	68.8	**70.6**	**70.6**	*(+0.0)*	89.6	92.1	**92.7**	*(+0.6)*	89.5	90.4	**91.1**	*(+0.7)*	73.7	79.8	**80.8**	*(+1.0)*
2000	73.6	**74.0**	**74.0**	*(+0.0)*	91.8	93.9	**94.1**	*(+0.2)*	90.9	91.7	**92.3**	*(+0.6)*	78.7	80.6	**82.2**	*(+1.6)*
4000	75.3	**83.8**	**83.8**	*(+0.0)*	91.8	**96.2**	**96.2**	*(+0.0)*	91.5	**93.9**	93.8	*(-0.1)*	83.4	92.9	**93.3**	*(+0.4)*

Table A4: Full Results (2 of 2): P@1 of Procrustes (P), SGM (S) or GOAT (G). Δ is gain/loss of GOAT over SGM.

	IP	*IS*	*IG*	*IP*	*IS*	*IG*	*IP*	*IS*	*IG*	*IP*	*IS*	*IG*	*IP*	*IS*	*IG*	*IP*	*IS*	*IG*
Seeds		en-bn			bn-en			en-bs			bs-en			en-de			de-en	
0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.1	0.1	0.0	0.3	**49.2**	0.1	0.3	**34.4**
25	0.6	**37.3**	*35.6*	3.2	**47.3**	*46.8*	0.2	**52.3**	*51.2*	0.1	**56.5**	*56.1*	**61.5**	49.8	49.7	**58.8**	47.6	47.2
50	1.6	**37.8**	*35.8*	24.3	**47.5**	*47.2*	48.3	**52.2**	*51.4*	4.2	**57.2**	*56.2*	**61.5**	49.9	49.6	**59.0**	47.8	47.9
75	**38.8**	37.3	36.7	**51.9**	48.3	47.6	**54.7**	52.3	51.5	**58.4**	57.1	56.4	**61.5**	50.4	49.9	**59.0**	48.5	47.9
100	**44.4**	37.7	36.5	**53.2**	48.2	47.9	**54.6**	52.5	51.1	**59.0**	57.1	56.8	**62.1**	50.2	50.0	**59.4**	48.4	48.1
200	**43.3**	38.9	37.2	**52.1**	48.0	48.1	**55.1**	52.9	52.4	**59.3**	58.8	57.7	**62.0**	50.9	50.4	**59.7**	49.3	48.8
500	**43.9**	40.3	39.5	**51.8**	49.7	48.8	53.7	**54.4**	53.0	59.8	**60.1**	59.7	**62.8**	52.3	52.0	**60.4**	50.2	50.1
1000	**43.4**	42.2	41.3	50.8	*51.1*	**51.4**	53.4	**55.9**	*55.0*	58.8	**61.4**	**61.4**	**63.5**	54.5	54.2	**61.4**	53.8	53.6
2000	42.8	**45.7**	*45.2*	49.8	**54.6**	*54.4*	52.3	**57.8**	*57.4*	59.8	**65.9**	*65.7*	**65.2**	61.8	61.2	**64.0**	60.5	60.4
4000	40.3	**60.3**	*58.8*	45.2	*65.9*	**66.6**	51.9	**70.4**	*69.9*	56.8	*83.7*	**86.4**	71.7	*74.1*	**74.4**	64.5	**71.4**	*71.2*
Seeds		en-et			et-en			en-fa			fa-en			en-id			id-en	
0	0.0	0.9	0.4	0.0	0.0	0.0	0.0	0.1	**45.2**	0.0	0.2	0.1	0.0	0.0	**15.5**	0.1	0.4	**58.2**
25	4.2	**51.8**	*51.3*	3.1	**66.4**	*65.9*	1.0	**46.7**	*45.4*	1.3	**47.6**	*46.7*	0.8	**55.9**	*54.8*	1.4	**59.5**	*58.5*
50	**59.3**	52.8	51.6	**66.3**	66.2	66.0	**52.9**	46.5	45.7	9.3	**47.7**	*47.0*	**64.7**	56.6	55.4	**65.4**	59.0	58.5
75	**58.8**	52.8	51.6	**66.6**	66.2	66.2	**53.1**	46.8	46.0	**54.0**	47.7	47.1	**64.9**	57.0	55.9	**65.8**	59.6	58.7
100	**59.3**	53.1	51.9	66.3	**66.6**	65.9	**53.0**	47.3	46.2	**54.1**	47.8	47.0	**65.2**	57.0	55.7	**66.0**	59.2	58.8
200	**59.0**	53.4	51.9	66.3	*66.5*	**66.7**	**53.3**	47.2	46.3	**54.2**	47.9	47.6	**64.9**	57.3	56.8	**66.5**	60.1	59.5
500	**59.4**	55.4	54.1	67.1	**68.5**	*68.0*	**53.3**	48.4	48.2	**54.3**	48.3	48.4	**66.4**	60.2	59.1	**67.4**	62.6	61.9
1000	**60.1**	58.3	57.7	67.6	*69.8*	**70.4**	**52.9**	49.8	49.9	**54.2**	49.8	49.4	**67.8**	63.6	62.6	**67.6**	64.2	64.0
2000	59.6	**64.2**	*63.2*	67.3	*74.1*	**74.2**	53.3	**54.8**	*54.6*	**54.5**	**54.5**	54.2	68.7	**69.6**	*69.2*	69.3	**70.2**	**70.2**
4000	61.7	**77.0**	**77.0**	67.0	**86.4**	**86.4**	52.1	**65.5**	**65.5**	53.4	**65.3**	*64.1*	72.1	**84.1**	*83.8*	72.8	*80.5*	**80.6**
Seeds		en-mk			mk-en			en-ms			ms-en			en-ru			ru-en	
0	0.0	0.1	0.0	0.0	0.0	0.1	0.0	0.1	0.2	0.0	0.0	0.1	0.0	**57.3**	0.1	0.0	1.1	**55.9**
25	1.0	**59.4**	*58.6*	1.0	**65.2**	*64.8*	0.7	**46.1**	*43.8*	0.7	**59.4**	*58.6*	**66.2**	57.7	57.0	**63.0**	55.9	55.5
50	**62.1**	59.2	58.6	**66.3**	65.6	65.4	**58.3**	46.4	43.7	**64.3**	59.4	58.7	**66.5**	58.2	57.3	**63.4**	56.0	56.0
75	**61.7**	59.1	59.1	**66.8**	65.4	65.5	**59.9**	47.0	45.3	**64.0**	58.7	58.5	**66.7**	58.3	57.6	**62.7**	55.9	56.1
100	**61.8**	59.3	59.3	**66.2**	65.6	65.5	**60.9**	47.7	45.6	**64.1**	59.5	58.7	**66.8**	57.8	57.6	**62.7**	56.3	55.6
200	**62.3**	60.0	59.2	**67.0**	66.0	65.7	**60.6**	48.5	47.2	**64.5**	59.5	58.9	**66.6**	58.9	57.8	**63.1**	56.4	56.5
500	**62.2**	60.5	60.6	66.7	**66.9**	66.5	**60.6**	52.6	51.0	**65.0**	61.4	61.1	**67.7**	60.0	59.3	**63.7**	58.0	58.0
1000	62.0	**62.5**	**62.5**	66.6	**68.0**	*67.9*	**61.1**	58.5	57.4	**65.0**	62.5	63.2	**68.3**	62.0	61.8	**64.0**	60.3	61.1
2000	61.8	**65.0**	*64.8*	66.1	*71.4*	**71.8**	60.4	**63.9**	*62.6*	64.9	**67.5**	**67.5**	**69.5**	67.2	67.6	65.7	*68.2*	**68.5**
4000	62.1	**75.7**	*75.6*	64.5	**88.4**	*88.0*	59.9	**79.7**	*79.5*	64.8	**77.8**	*77.5*	74.5	*81.9*	**82.4**	69.0	**89.3**	**89.3**
Seeds		en-ta			ta-en			en-vi			vi-en			en-zh			zh-en	
0	0.0	0.0	0.0	0.0	0.2	**0.9**	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.3	0.0	0.0	0.1	0.1
25	0.3	**5.8**	*4.3*	0.9	*45.8*	**46.1**	0.1	0.3	0.3	0.2	*23.5*	**44.1**	3.3	**13.7**	*7.3*	2.0	**14.6**	*7.9*
50	5.5	**33.6**	*32.5*	18.0	**46.6**	*46.4*	0.3	*22.0*	**42.6**	1.0	**48.9**	*46.3*	**52.9**	19.0	12.3	**52.5**	17.2	11.6
75	**38.3**	33.8	32.4	**50.0**	46.9	46.2	44.6	**46.5**	43.3	8.4	**49.4**	*48.7*	**53.5**	19.7	13.7	**51.9**	17.0	12.9
100	**39.5**	33.3	32.5	**50.5**	47.2	46.8	3.4	**47.8**	*43.8*	**59.4**	49.6	48.8	**53.2**	20.5	13.1	**52.1**	18.3	13.4
200	**40.2**	34.0	32.8	**50.8**	47.0	47.1	**57.8**	49.1	45.9	**60.1**	50.9	49.7	**53.4**	22.4	15.0	**52.6**	20.2	15.0
500	**40.0**	35.4	34.0	**50.0**	49.0	48.1	**59.2**	52.8	50.6	**61.4**	54.5	53.1	**53.3**	29.3	22.1	**52.8**	28.3	24.4
1000	**38.2**	37.0	36.0	49.7	**50.5**	*50.3*	**59.7**	57.1	56.4	**64.3**	58.8	58.9	**54.2**	34.1	29.6	**54.3**	34.4	30.8
2000	38.2	**40.7**	*38.8*	49.3	**54.3**	*53.8*	60.6	*61.5*	**62.1**	**68.0**	65.8	66.3	**56.0**	44.5	42.4	**54.5**	42.2	41.3
4000	34.1	*48.8*	**49.6**	48.9	*72.9*	**73.3**	62.5	*77.7*	**78.3**	73.2	**82.3**	**82.3**	59.1	**67.2**	*66.9*	58.4	**66.4**	*65.8*

Table A5: P@1 of Iterative Procrustes (IP), Iterative SGM (IS), and Iterative GOAT (IG). **Highest per row**. IterSGM/IterGOAT are *italicized* when outperforming IterProc but are not highest in row.

	en-to-*					*-to-en					en-to-*					*-to-en				
		SGM		GOAT			SGM		GOAT			SGM		GOAT			SGM		GOAT	
Seeds	*Prev*	*-PP*	*-PS*	*-PP*	*-PG*	*Prev*	*-PP*	*-PS*	*-PP*	*-PG*	*Prev*	*-PP*	*-PS*	*-PP*	*-PG*	*Prev*	*-PP*	*-PS*	*-PP*	*-PG*
	bn										mk									
0	*0.1*	0.0	0.1	0.0	0.1	*0.1*	0.0	0.0	0.1	0.0	*0.1*	0.0	0.1	0.0	0.0	*0.1*	0.1	0.0	0.0	0.1
25	*37.3*	44.0	0.5	**44.3**	0.5	*47.3*	52.8	8.4	**53.1**	6.2	*59.4*	**64.4**	1.2	63.9	8.0	*65.2*	68.7	1.1	**68.8**	0.6
50	*37.8*	43.1	2.0	**44.3**	3.1	*47.5*	52.1	14.4	**53.0**	48.2	*62.1*	**64.0**	63.6	63.9	62.8	*66.3*	68.4	68.1	**68.5**	67.8
75	*38.8*	**44.7**	38.8	**44.7**	39.5	*51.9*	53.2	49.2	**53.5**	49.1	*61.7*	63.8	63.7	**64.3**	63.3	*66.8*	**69.2**	67.3	68.8	67.9
100	*44.4*	44.3	40.4	**45.3**	39.7	*53.2*	52.4	48.5	**53.9**	48.1	*61.8*	64.4	**65.1**	64.1	64.2	*66.2*	69.3	68.2	**69.4**	67.5
200	*43.3*	**45.4**	40.9	45.2	40.8	*52.1*	53.8	50.4	**54.0**	50.6	*62.3*	64.1	64.5	**64.9**	64.2	*67.0*	69.5	67.8	**69.7**	68.0
500	*43.9*	**46.8**	43.9	**46.8**	43.2	*51.8*	54.3	50.3	**55.1**	51.4	*62.2*	65.3	**65.7**	65.3	65.5	*66.9*	69.6	69.1	**70.0**	68.9
1000	*43.4*	**47.1**	45.4	**47.1**	45.3	*51.5*	55.3	53.2	**55.5**	53.9	*62.5*	65.8	**66.5**	65.5	**66.5**	*68.0*	69.8	**69.9**	**69.9**	69.5
2000	*45.9*	48.9	**49.3**	48.7	**49.3**	*55.2*	56.6	**57.2**	56.6	56.3	*65.0*	67.0	**68.2**	66.8	67.9	*71.8*	71.2	72.5	71.0	**73.1**
4000	*60.3*	52.0	**61.2**	50.5	**61.2**	*66.6*	55.2	**68.9**	55.2	**68.9**	*75.7*	69.0	**77.2**	68.9	77.1	*88.8*	74.1	**91.1**	74.1	**91.1**
	bs										ms									
0	*0.0*	0.0	0.0	0.0	0.0	*0.1*	0.0	0.0	0.0	0.0	*0.2*	0.0	0.0	0.0	0.0	*0.1*	0.0	0.0	0.1	0.0
25	*52.3*	57.4	0.2	**58.4**	0.4	*56.5*	**61.7**	0.8	61.5	0.6	*46.1*	62.1	0.4	**62.6**	0.9	*59.4*	**65.6**	0.5	65.3	0.2
50	*52.2*	**57.4**	25.7	**57.4**	53.8	*57.2*	61.2	6.0	**61.9**	42.3	*58.3*	61.8	6.4	**62.7**	58.8	*64.3*	**65.6**	63.0	**65.6**	63.1
75	*54.7*	57.5	54.6	**58.1**	55.1	*58.4*	61.6	57.9	**61.7**	57.2	*59.9*	62.4	59.0	**62.6**	59.8	*64.0*	65.5	64.0	**65.6**	62.8
100	*54.6*	57.4	55.0	**58.1**	55.4	*59.0*	**61.8**	59.0	61.3	57.5	*60.9*	62.6	59.7	**63.0**	58.6	*64.1*	**66.0**	63.9	65.8	63.1
200	*55.1*	57.8	56.9	**58.6**	56.4	*59.3*	61.9	58.9	**62.4**	58.3	*60.6*	**63.1**	60.6	**63.1**	59.6	*64.5*	**66.7**	64.3	66.4	63.9
500	*54.4*	58.5	57.1	**59.1**	56.7	*60.1*	63.1	61.1	**63.8**	61.0	*60.6*	63.4	61.1	**64.1**	61.0	*65.0*	**67.4**	65.9	66.8	65.8
1000	*55.9*	**59.8**	59.1	59.6	58.7	*61.6*	63.6	63.4	**64.7**	62.9	*61.1*	65.9	64.8	**66.0**	64.4	*65.0*	67.3	67.0	**67.7**	66.9
2000	*58.1*	59.2	**60.7**	59.9	60.5	*65.9*	66.9	68.8	66.6	**69.1**	*65.0*	67.4	**68.8**	67.0	68.5	*67.5*	69.1	**69.8**	69.4	69.7
4000	*70.6*	63.0	**72.2**	63.4	71.8	***86.4***	69.7	84.7	69.7	85.7	***79.7***	70.9	**79.7**	70.7	79.5	*77.8*	70.3	**79.2**	70.0	79.1
	de										ru									
0	*49.2*	0.1	0.1	**61.8**	0.0	*34.5*	0.5	0.2	**59.2**	0.3	***57.3***	0.0	0.0	0.1	0.1	***55.9***	0.0	0.0	4.4	0.0
25	*61.5*	**61.9**	59.3	61.7	59.1	*58.8*	**59.5**	56.7	59.3	56.8	*66.2*	**67.8**	66.3	67.7	66.1	*63.0*	**64.4**	62.6	63.9	62.0
50	*61.5*	**62.1**	59.9	**62.1**	59.3	*59.0*	**59.6**	56.5	59.1	56.5	*66.5*	**67.6**	66.0	67.5	66.5	*63.4*	**64.4**	62.5	63.7	61.5
75	*61.5*	62.0	59.6	**62.3**	59.7	*59.0*	**59.5**	56.6	59.4	57.1	*66.7*	67.8	66.6	**68.1**	67.0	*62.7*	**64.2**	62.6	63.9	61.6
100	*62.1*	**62.3**	60.1	62.2	59.7	*59.4*	**59.5**	57.1	59.4	56.8	*66.8*	**68.3**	67.2	67.9	66.4	*62.7*	**64.0**	61.6	63.8	61.3
200	*62.0*	**62.5**	60.7	62.4	60.3	*59.7*	**60.0**	58.1	**60.0**	57.6	*66.6*	**68.7**	67.5	68.6	67.1	*63.1*	**64.5**	62.2	64.4	61.9
500	*62.8*	**63.8**	61.9	**63.8**	61.8	*60.4*	**61.1**	59.3	61.0	59.5	*67.7*	**69.0**	68.8	68.9	68.2	*63.7*	**65.0**	63.8	64.8	64.1
1000	*63.5*	**64.1**	63.0	**64.1**	63.1	*61.4*	**62.3**	61.4	**62.3**	61.9	*68.3*	70.3	69.9	**70.4**	69.8	*64.0*	66.5	**67.1**	66.7	66.5
2000	*65.2*	66.6	**69.6**	66.3	69.4	*64.0*	65.4	**68.2**	65.1	67.9	*69.5*	72.2	74.0	72.6	**74.2**	*68.5*	69.5	**72.7**	69.3	72.5
4000	*74.4*	73.2	**79.7**	72.9	79.5	*71.4*	67.0	**77.6**	67.2	**77.6**	*83.3*	78.3	**86.5**	79.3	**86.5**	*89.3*	77.4	**89.3**	77.4	**89.3**
	et										ta									
0	*0.9*	0.0	0.0	0.2	0.0	*0.1*	0.0	0.0	0.0	0.0	*0.0*	0.0	0.0	0.0	0.1	*0.9*	0.1	0.0	0.6	0.0
25	*51.8*	60.0	2.6	**60.4**	5.9	*66.4*	69.4	8.9	**70.2**	15.0	***5.8***	2.1	0.5	2.2	0.6	*46.1*	51.0	2.0	**51.4**	2.4
50	*59.3*	**61.0**	59.2	60.9	58.4	*66.3*	69.4	66.2	**70.2**	66.6	*33.6*	**40.5**	2.1	40.2	10.5	*46.6*	50.7	47.9	**52.0**	46.3
75	*58.8*	**60.7**	58.7	60.6	58.6	*66.6*	68.8	66.4	**69.8**	66.5	*38.3*	40.0	33.5	**40.4**	35.7	*50.0*	52.0	47.6	**52.4**	46.9
100	*59.3*	61.0	58.3	**61.1**	59.0	*66.6*	69.8	66.7	**70.2**	66.7	*39.5*	**40.5**	35.2	40.2	36.8	*50.5*	51.5	48.2	**52.5**	47.9
200	*59.0*	61.3	60.4	**61.5**	59.9	*66.7*	70.2	67.7	**70.5**	67.3	*40.2*	40.3	36.6	**40.5**	36.2	*50.8*	52.2	48.1	**52.9**	49.3
500	*59.4*	62.9	61.3	**63.2**	60.3	*68.5*	70.8	69.7	**71.5**	69.0	*40.0*	**41.5**	38.7	40.9	39.1	*50.0*	**53.0**	51.4	**53.0**	50.9
1000	*60.1*	**64.3**	63.9	**64.3**	63.4	*70.4*	**72.5**	69.9	**72.5**	70.1	*38.2*	41.3	**41.7**	41.6	40.7	*50.5*	53.0	52.3	**53.7**	51.2
2000	*64.2*	66.0	**67.7**	66.6	67.4	*74.2*	73.5	74.8	74.1	**75.0**	*40.7*	43.5	43.7	42.7	**44.2**	*55.1*	55.6	**56.6**	55.3	56.2
4000	*77.4*	72.5	**80.2**	71.7	**80.2**	*86.4*	80.7	**87.2**	80.7	**87.2**	*49.6*	43.7	**52.4**	44.0	51.7	*73.8*	64.4	**71.6**	65.3	**71.6**
	fa										vi									
0	***45.2***	0.1	0.0	0.5	0.0	*0.2*	0.2	0.0	0.1	0.0	*0.1*	0.0	0.1	0.0	0.1	*0.1*	0.1	0.4	0.1	0.9
25	*46.7*	54.3	1.0	**54.4**	4.5	*47.6*	**55.1**	1.8	**55.1**	2.0	*0.4*	0.4	0.1	0.4	0.2	***44.1***	1.3	0.3	5.3	0.2
50	*52.9*	**54.0**	51.6	53.8	51.2	*47.7*	**55.2**	51.6	54.9	52.3	***42.6***	2.4	0.2	5.2	0.4	*48.9*	58.8	1.2	**59.1**	2.1
75	*53.1*	**54.5**	52.0	54.1	51.9	*54.0*	**55.4**	52.7	55.3	52.0	*46.5*	54.5	8.8	**55.0**	1.2	*49.4*	59.3	28.0	**59.6**	54.4
100	*53.0*	54.2	52.5	**54.3**	52.3	*54.1*	55.0	52.5	**55.5**	52.7	*47.8*	55.4	22.3	**55.6**	36.2	***59.4***	59.1	56.6	59.3	56.3
200	*53.3*	54.3	52.3	**54.8**	52.2	*54.2*	**55.2**	53.0	54.6	51.9	*57.8*	**58.0**	55.4	57.8	56.0	*60.1*	60.7	58.7	**61.2**	57.7
500	*53.3*	**55.5**	53.4	**55.5**	53.1	*54.3*	55.7	53.4	**56.1**	52.7	*59.2*	59.6	59.0	**60.0**	58.3	*61.4*	63.4	61.9	**63.5**	60.8
1000	*52.9*	56.0	54.8	**56.3**	54.8	*54.2*	**55.7**	54.2	55.6	54.1	*59.7*	63.5	61.2	**63.6**	61.2	*64.3*	67.1	65.8	**68.1**	65.3
2000	*54.8*	57.7	**58.6**	58.0	58.4	*54.5*	56.9	**57.7**	57.5	56.9	*62.1*	66.7	66.7	**67.3**	65.8	*68.0*	73.2	71.8	**73.5**	71.5
4000	*65.9*	62.9	67.2	62.4	**67.4**	*65.5*	61.0	**67.7**	60.1	67.0	*78.3*	72.8	80.3	73.0	**80.5**	*82.3*	81.0	**84.6**	80.1	84.3
	id										zh									
0	***15.5***	0.6	0.0	4.4	0.0	***58.2***	0.0	0.0	1.2	0.0	*0.3*	0.2	0.0	0.1	0.0	*0.1*	0.0	0.0	0.0	0.0
25	*55.9*	**65.9**	1.4	65.8	0.6	*59.5*	**66.8**	0.4	66.7	1.8	*13.7*	52.2	1.7	**52.7**	1.7	*14.6*	**51.7**	0.8	48.1	0.8
50	*64.7*	65.8	63.4	**66.0**	63.2	*65.4*	66.6	64.4	**67.0**	64.0	***52.9***	52.6	47.6	52.6	46.2	***52.5***	51.7	48.0	50.5	47.3
75	*64.9*	**66.1**	64.0	66.0	63.3	*65.8*	66.6	64.1	**66.7**	64.2	***53.5***	52.3	48.3	51.6	46.4	***51.9***	51.2	48.5	51.1	48.0
100	*65.2*	**66.6**	63.6	66.2	63.2	*66.0*	67.0	64.4	**67.1**	63.9	***53.2***	51.9	49.4	51.6	47.3	***52.1***	51.1	48.1	51.0	48.0
200	*64.9*	66.7	64.8	**66.8**	65.0	*66.5*	**67.4**	64.9	67.3	65.0	***53.4***	52.7	49.6	52.6	49.2	***52.6***	51.6	49.3	51.4	48.9
500	*66.4*	**68.0**	66.8	**68.0**	66.7	*67.4*	68.9	67.1	**69.1**	67.2	*53.3*	**54.0**	52.0	53.1	51.1	*52.8*	**53.0**	50.9	52.6	50.4
1000	*67.8*	69.9	69.8	**70.1**	69.7	*67.6*	**70.0**	68.9	**70.0**	68.3	*54.2*	54.9	53.2	**55.4**	52.2	*54.3*	**54.7**	53.0	54.3	52.0
2000	*70.7*	72.2	**74.9**	72.1	74.2	*70.2*	72.2	**73.0**	72.5	72.9	*56.0*	**59.2**	59.1	58.0	57.7	*54.5*	57.6	**57.8**	57.6	56.8
4000	*84.3*	77.1	**86.2**	76.8	**86.2**	*80.6*	78.0	**84.0**	78.2	83.7	*67.5*	66.9	74.5	66.4	**75.1**	*66.4*	65.8	72.9	65.3	**73.3**

Table A6: Full Results: P@1 of Combination Experiments. SGM-PP starts with SGM, ends with Procrustes. SGM-PS: IterProc then SGM. GOAT-PP: start GOAT, end Pro[illegible]GOAT-PG: IterProc then GOAT. Previous best of all other experiments is in the *Prev* column. *Prev* here includes iterative results from Table A5.