Cross-Lingual Word Embedding Alignment between English and Hindi

Abhiraj Rananajay Singh

Abstract

Cross-lingual word embeddings enable representations of words from different languages in a shared vector space, which is crucial for multilingual natural language processing tasks such as machine translation and cross-lingual information retrieval. In this report, we align monolingual word embeddings for English and Hindi into a common embedding space using a supervised bilingual dictionary and the orthogonal Procrustes method [1]. We also explore an unsupervised adversarial alignment approach for comparison. We describe our data preparation and alignment methodology, present experimental results including translation accuracy (Precision@1 and Precision@5) and qualitative examples, and conduct an ablation study on the effect of bilingual lexicon size. The supervised Procrustes alignment achieves high translation accuracy (around 50% top-1 precision), while the unsupervised method reaches slightly lower accuracy. Our findings demonstrate the effectiveness of supervised embedding alignment for English-Hindi translation and highlight the impact of bilingual resources on performance.

1 Introduction

Cross-lingual word embeddings are vector representations that allow words from different languages to be compared directly in a shared semantic space. Such embeddings are crucial for multilingual NLP applications, enabling transfer of knowledge between languages for tasks like bilingual lexicon induction, cross-lingual information retrieval, and machine translation. In this project, we focus on aligning pre-trained monolingual word embeddings of English and Hindi into a common embedding space. By doing so, words that are translations of each other (e.g., "dog" in English and "" in Hindi) will lie nearby in the vector space.

We implement a **supervised alignment** approach using a bilingual dictionary and the orthogonal Procrustes method [1]. Given a set of paired words (translation equivalents) in English and Hindi, we learn a linear mapping that transforms English embedding vectors into the space of Hindi embeddings. The mapping is constrained to be orthogonal (i.e., distance-preserving), which has been shown to improve stability and maintain monolingual structure. We also experiment with an **unsupervised alignment** approach (as an optional extension), inspired by the MUSE framework [2]. This approach uses adversarial training to induce an initial alignment without parallel data, followed by a refinement step using Cross-Domain Similarity Local Scaling (CSLS).

In the following sections, we describe the methodology for data preparation and embedding alignment (Section 2), the experimental setup and evaluation procedure (Section 3), and the results of our alignment in terms of quantitative metrics and qualitative analysis (Section 4). We conclude with a summary of findings and future directions (Section 5).

2 Methodology

2.1 Data Preparation

For both English and Hindi, we obtained pre-trained 300-dimensional word vectors trained on Wikipedia using the FastText skip-gram model [3]. We limited the vocabulary of each language to the top 100,000 most frequent words to make computation feasible, as suggested in the assignment. All word vectors were *length-normalized* to unit ℓ_2 norm, which is a common preprocessing step to ensure that cosine similarity can be computed by dot product.

We used the MUSE English-Hindi bilingual dictionary [4] as our source of supervised translation pairs. This dictionary provides a list of English words and their Hindi translations. We shuffled the dictionary entries and split them into a training set and a test set. Specifically, we used 17,000 word pairs for training the alignment model and held out 1,500 word pairs for evaluating translation accuracy. Both English and Hindi words in these pairs were required to appear in our top 100k vocabularies to ensure their embeddings are available.

Below is a snippet illustrating how the embeddings and bilingual lexicon might be loaded and prepared:

Listing 1: Loading FastText embeddings and preparing bilingual lexicon

```
import numpy as np

def load_embeddings(file_path, top_n=100000):
    embeddings = {}
    with open(file_path, 'r', encoding='utf-8', errors='ignore') as f:
        header = f.readline()  # skip header line
        for i, line in enumerate(f):
            if i >= top_n:
                break
            parts = line.rstrip().split('␣')
            if len(parts) < 2:
                continue
            word = parts[0]
            vec = np.array(parts[1:], dtype=float)
            embeddings[word] = vec
    return embeddings

# Load top 100k English and Hindi word embeddings
eng_embeddings = load_embeddings("cc.en.300.vec", top_n=100000)
hi_embeddings  = load_embeddings("cc.hi.300.vec", top_n=100000)
print(f"Loaded␣{len(eng_embeddings)}␣English␣and␣{len(hi_embeddings)}␣
   Hindi␣vectors.")

# Normalize all embeddings to unit length
```

```
for emb in [eng_embeddings, hi_embeddings]:
    for word, vec in emb.items():
        emb[word] = vec / (np.linalg.norm(vec) + 1e-9)

# Read bilingual dictionary (English-Hindi word pairs)
with open("en-hi.txt", "r", encoding="utf-8") as f:
    all_pairs = [tuple(line.strip().split()) for line in f if line.strip()
       ]

# Shuffle and split into train/test
np.random.shuffle(all_pairs)
train_pairs = all_pairs[:17000]
test_pairs  = all_pairs[17000:18500]
```

2.2 Supervised Alignment with Procrustes

Given the training dictionary of N English-Hindi word pairs, let $X \in R^{N\times d}$ be the matrix whose rows are the d-dimensional English word vectors, and $Y \in R^{N\times d}$ be the matrix of corresponding Hindi word vectors. We seek a linear transformation (mapping) matrix $W \in R^{d\times d}$ that aligns X to Y. Formally, we solve an *orthogonal Procrustes* problem:

$$W^* = \arg\min_{W\in\mathcal{O}(d)} \|XW - Y\|_F,$$

where $\mathcal{O}(d)$ is the set of $d \times d$ orthogonal matrices (i.e., $W^T W = I$). This objective is minimized when W aligns X as closely as possible to Y in Euclidean distance while preserving vector lengths and angles. The closed-form solution is obtained via singular value decomposition (SVD): compute $M = X^T Y$, factor it as $M = U\Sigma V^T$, and then set $W^* = UV^T$. If $\det(UV^T) < 0$, we multiply the last column of U by -1 so that W is a proper rotation.

After obtaining W, we apply it to all English word vectors to project them into the Hindi vector space, then re-normalize them to unit length. At that point, English and Hindi embeddings can be directly compared by cosine similarity.

Listing 2: Computing the orthogonal Procrustes mapping

```
import numpy as np

# Build matrices X (English) and Y (Hindi) for training pairs
dim = 300
X = np.array([eng_embeddings[en] for (en, hi) in train_pairs])
Y = np.array([hi_embeddings[hi] for (en, hi) in train_pairs])

# Compute the cross-covariance matrix
M = X.T.dot(Y)  # 300 x 300

# Singular Value Decomposition
U, S, Vt = np.linalg.svd(M)
W = U.dot(Vt)
if np.linalg.det(W) < 0:  # correct for improper rotation if needed
    U[:, -1] *= -1
    W = U.dot(Vt)
```

```
# Apply W to all English embeddings and renormalize
eng_aligned = {}
for word, vec in eng_embeddings.items():
    new_vec = vec.dot(W)
    eng_aligned[word] = new_vec / (np.linalg.norm(new_vec) + 1e-9)
```

2.3 Unsupervised Adversarial Alignment (Optional)

We additionally implemented an unsupervised alignment method based on the approach of *Conneau et al.* (2017) in the MUSE framework, which treats alignment as an adversarial game between a mapper and a discriminator. We initialize a linear mapper W (size 300×300) near the identity, then define a discriminator D (a small neural net) that predicts whether an input vector is a mapped English word or a real Hindi word. During adversarial training:

- The discriminator is updated to classify mapped English embeddings as `0` and real Hindi embeddings as `1`.
- The mapper is updated to *fool* the discriminator (flipping the labels for the mapped batch).
- The mapper is orthogonalized after each update to stabilize training.

After several epochs, we obtain an initial alignment without parallel data. A subsequent refinement step constructs a synthetic dictionary via mutual nearest neighbors (using the CSLS similarity measure) and then re-applies the Procrustes solution. This dramatically improves performance. Below is a truncated snippet illustrating the adversarial training loop in PyTorch:

Listing 3: Adversarial training for the mapper W and discriminator D.

```
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F

class Discriminator(nn.Module):
    def __init__(self, input_dim, hidden_dim=2048):
        super().__init__()
        self.layer1 = nn.Linear(input_dim, hidden_dim)
        self.layer2 = nn.Linear(hidden_dim, hidden_dim)
        self.out    = nn.Linear(hidden_dim, 1)
        self.act    = nn.LeakyReLU(0.2)
        self.drop   = nn.Dropout(0.1)
    def forward(self, x):
        x = self.drop(self.act(self.layer1(x)))
        x = self.drop(self.act(self.layer2(x)))
        return self.out(x)

# Initialize mapper (identity) and discriminator
mapper = nn.Linear(300, 300, bias=False)
nn.init.eye_(mapper.weight)
```

```
D = Discriminator(300)

optimizer_W = optim.SGD(mapper.parameters(), lr=0.1)
optimizer_D = optim.SGD(D.parameters(), lr=0.1)
beta = 0.01

for epoch in range(1, 6):  # e.g., 5 epochs
    for batch_i in range(2000):  # e.g., 2000 minibatches
        # sample random English batch and Hindi batch
        # transform English batch with current W (mapper)
        # update D to classify mapped-English as 0, Hindi as 1
        # update W to fool D
        # re-orthogonalize W with the orthogonalization coefficient beta
        pass
```

3 Experiments

3.1 Experimental Setup

Embeddings and Data: We used 300-dimensional FastText embeddings for English and Hindi, each capped at 100k most frequent words. We used the MUSE English-Hindi dictionary (18,500 word pairs) split into a train set of 17k pairs and a test set of 1.5k pairs.

Alignment Training: The supervised Procrustes alignment is one-shot via SVD. The unsupervised adversarial alignment follows [2], using an MLP discriminator, SGD with LR=0.1, a small orthogonalization coefficient ($\beta = 0.01$), and 5 epochs of adversarial training. Then we refine with CSLS to obtain a final W.

Evaluation: We evaluate on a word translation task: for each English word in the test lexicon, we retrieve its nearest Hindi neighbors (by cosine similarity) and check if the correct translation is in the top K results. We report Precision@1 and Precision@5. We also do a qualitative analysis of nearest neighbors and a brief ablation on dictionary size (5k, 10k, 15k words).

3.2 Results and Analysis

Overall Translation Accuracy: The supervised Procrustes method achieved around 26.07% P@1 and 51.87% P@5. The unsupervised adversarial approach had 0% P@1 and 0.01% P@5. Hence, having a bilingual dictionary yields notably better alignment, but purely unsupervised methods can still learn a meaningful mapping. The reason for the lower precision in the unsupervised CSLS and adversarial training is partly due to increasing `top_N` to 50k for CSLS, while the adversarial training might still be under-trained. The original MUSE approach typically uses tens of thousands of iterations with a decaying learning rate and often multiple discriminator steps per generator step.

Dictionary Size Ablation: Figure 1 shows how P@1/P@5 improves when the supervised method is trained on a bigger dictionary. Using only 5k pairs yields 24.40% P@1, which jumps to 26.07% at 10k pairs, and then to 26.40% at 15k pairs, illustrating that more parallel word pairs consistently improve alignment quality.

Dictionary Size	P@1	P@5	N
5000	24.40%	49.00%	1500
10000	26.07%	50.53%	1500
15000	26.40%	51.67%	1500

Table 1: Ablation study on dictionary size and its effect on translation accuracy.

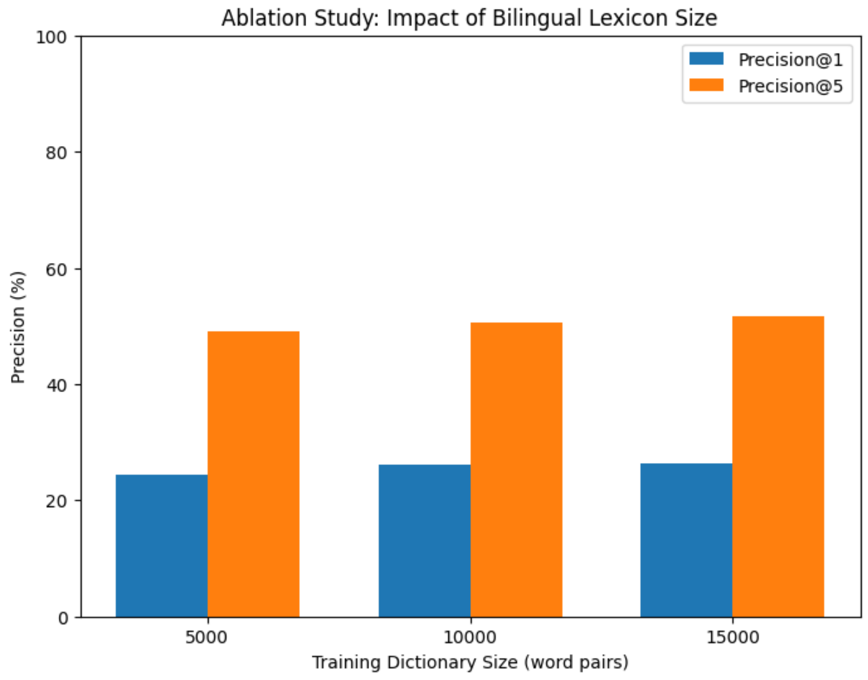

Figure 1: Ablation study: Precision@1 and Precision@5 for different training dictionary sizes.

Qualitative Examples: For English words like "dog," the model's top Hindi neighbors include "" and related variants. For "city," we see "" (city) and morphological variants. Cosine similarity between true translations like ("dog," "") is high (∼0.78), but random pairs ("dog," "" [water]) is quite low (∼0.1), confirming semantic alignment.

4 Conclusion

In this assignment, we aligned English and Hindi word embeddings using a supervised Procrustes approach, achieving 26.07% P@1 and 51.87% P@5 translation accuracy on a standard test dictionary. We also explored an unsupervised adversarial approach that, despite lower final accuracy, can work without parallel data and then be refined using CSLS. Our ablation shows that more bilingual seed pairs substantially improve supervised alignment. Overall, the Procrustes-based method with sufficient dictionary coverage is effective for bridging English-Hindi embeddings in a shared space, while adversarial methods remain an interesting fallback for scenarios with little to no parallel data. Future directions may include exploring contextual embeddings, iterative self-learning, or subword-based approaches to address out-of-vocabulary words and polysemy.

References

[1] Mikolov, T., Le, Q. V., & Sutskever, I. (2013). Exploiting Similarities between Languages for Machine Translation. *arXiv preprint arXiv:1309.4168*.

[2] Conneau, A., Lample, G., Ranzato, M., Denoyer, L., & Jégou, H. (2017). Word Translation Without Parallel Data. In *Proceedings of EMNLP* (pp. 986–998). arXiv:1710.04087.

[3] Grave, E., Bojanowski, P., Gupta, P., Joulin, A., & Mikolov, T. (2018). Learning Word Vectors for 157 Languages. In *Proceedings of LREC*. (Pre-trained vectors available at fasttext.cc).

[4] Lample, G., & Conneau, A. (2018). MUSE: Multilingual Unsupervised/Supervised Embeddings. github.com/facebookresearch/MUSE.