Sarvam - LLM Interview Assignment

Arjun Pawar *

7 March 2025

This write-up outlines my approach and describes the way my code is organized so that it can be reproduced. (Link to Colab notebook for reference is here)

1 Word embeddings

I pulled a dataset of 10k Wikipedia articles from Huggingface `datasets` and tried training word embeddings using the `fasttext` library and the `gensim` library and included this approach in the notebook. However, due to computational constraints I could not train on the full corpus. Hence, as per the instructions, I resorted to pre-trained FastText vectors (dim= 300) for Hindi and English. I also made pickle files which can be loaded to reproduce this.

2 Data preparation

I took a subset of the 100k most frequent words from each language and restricted the vocabulary to this. Using the MUSE en to hi translation dataset, I loaded the training and test translations which they had already split for us. The test dataset was kept separate and not included in any of the training.
Next, I did a sanity check to see how much of the training set came from the vocabulary for the project. This was done so that any training pairs that I use have valid word vectors which exist in the vocabulary of 100k words.

3 findMapping()

I wrote this function to find a mapping matrix, W for the Procrustes alignment. Following the reference paper (Conneau et al, 2018) I implemented this function which returns the optimal orthogonal matrix.

$$W = UV^T$$

where $U\Sigma V^T = SVD(YX^T)$

*arjun7@ucla.edu

4 findTranslations()

This function translates the given English vectors into Hindi words by first transforming the input vectors using the Procrustes matrix and then finding the k nearest neighbours based on regular cosine similarity. It returns a list of k Hindi words for each English vector.

5 calculateMetrics()

This function returns various metrics to evaluate the translations, since we have ground truth translations in the supervised case. The assignment asks to report precision but I divided the metrics into 3 as follows:

1. Top k accuracy: I computed this by checking if any one predicted translation lies in the set of expected translations.

2. Precision @k : This is found by counting number of relevant translations and dividing by k. I found a small issue with this method that some English words in the MUSE dataset had 1 Hindi translation, some had 2, some had 4 and so on... This causes a problem as Precision @k divides by k irrespective of how many ground truth translations exist. So I modified the task slightly by using another metric to take care of this.

3. Modified precision @k : I modified the previous definition so instead of dividing by k this divides by $\min(k, g)$ where g = number of Hindi translations for that word in the given dataset. This would be closer to the notion of precision@k.

6 Testing

6.1 Varying k and training size

I tested training lexicon of sizes 5k, 6k, 7k, 8k since we were capped by how much training data we had (so 10k+ pairs data was not available in the vocab).
I tested k values of $\{1, 3, 5, 7, 10\}$.
The results were as follows

As expected, the general trend was that model performance (accuracy and modified precision) increased with an increase in training size for a given k.

6.2 Observing cosine similarities

To test semantic similarities, I plotted box plots of similarity scores between English and Hindi vectors from the MUSE dataset and then the similarity scores between Transformed English (so predicted Hindi) and actual Hindi vectors. These can be seen in the notebook and show that the distribution of the data shifted much higher after we applied Procrustes alignment with the median similarity increasing from 0 to 0.5

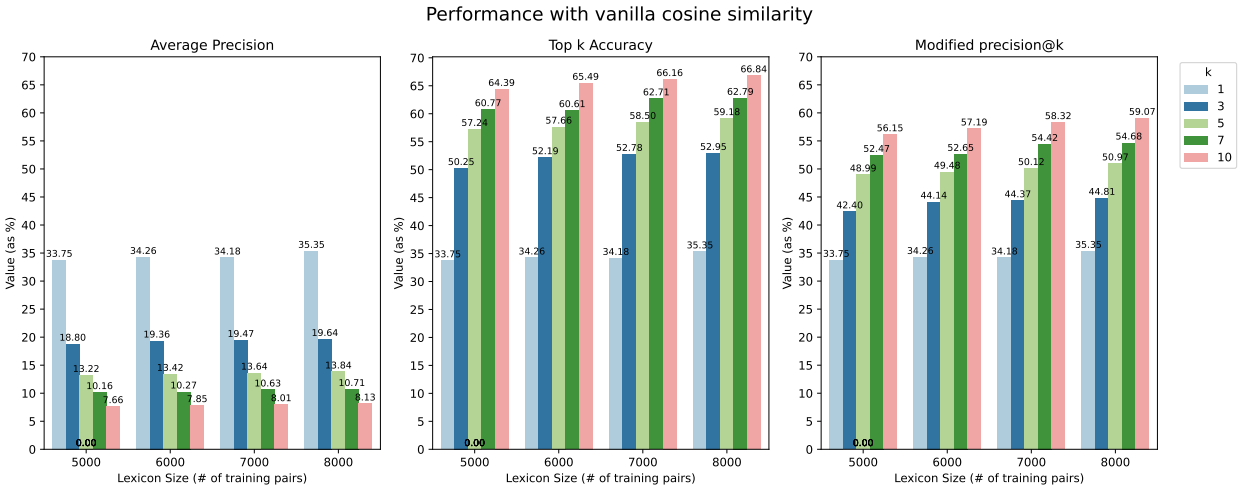

Figure 1

7 Extra credit: CSLS

I also implemented CSLS or Cross Domain Similarity Local Scaling which was proposed in the paper as a way to mitigate 'hubness'. This was used as an alternative to vanilla cosine similarity and I trained and tested again using this metric. The results are shown below.

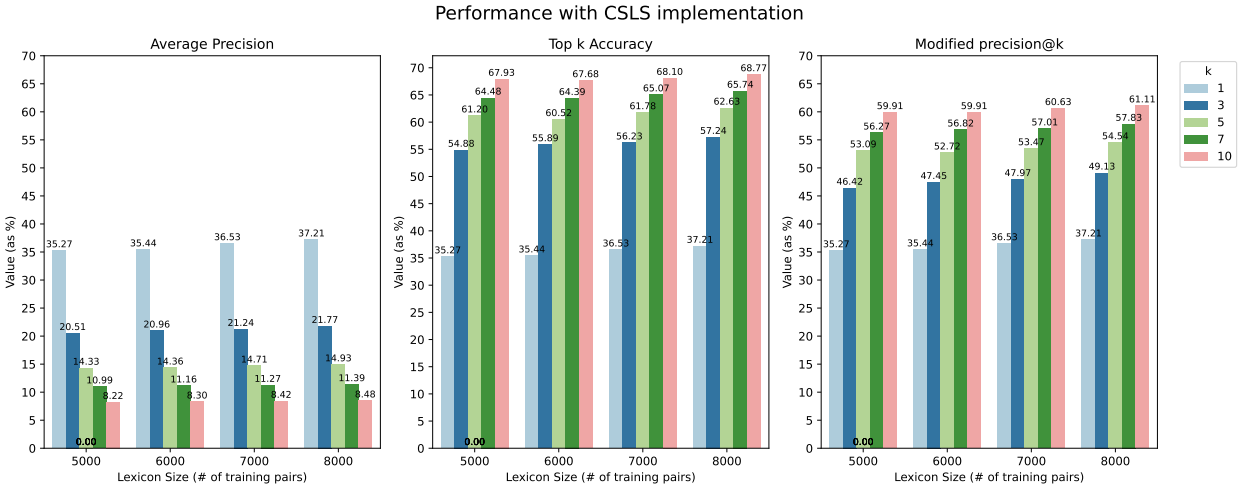

Figure 2

The visualization supports the previous trend of k and training size positively impacting the model performance metrics. However, now we see that using CSLS gives even **better** results with top10 accuracy for 8k pairs training size reaching 69% on the test set.

8 Extra credit: Unsupervised adversarial training

I was interested in implementing the pipeline for unsupervised adversarial learning as described in the paper. I laid out the architecture and setup based on how I would proceed, but did not train it fully due

to computational time/resource constraints. I used the following facts from the paper for the notebook implementation:

- discriminator: dropout rate 0.1 , multilayer perceptron with two hidden layers of size 2048 and Leaky-ReLU activation functions
- smoothing coefficient = 0.2, SGD with a batch size of 32, a learning rate of 0.1 and a decay of 0.95 both for the discriminator and W
- alternate the update of our model with the following update rule on the matrix W (eqn (7) in the paper)

The implementation of this architecture can be found in the notebook as well.

9 Conclusion

This investigation introduced me to a new technique, Procrustes alignment, and the paper offered a view into supervised vs unsupervised methods to align embedding spaces for translation tasks. Additionally, this method also considers polysemy which can improve translation performance. This could potentially help with low-resource Indic languages as we can leverage data and embeddings from well-documented languages to learn better transformation matrices that could translate language. Further deep dives into the semantics of translation could include looking at how phrase-level or sentence-level meanings are handled when words compose with one another, or how different stems and affixes of word are handled by this method.