Precog Task: Language Representations

Trisanu Bhar

September 18, 2025

Contents

1 Introduction

Here, we shall be exploring the experiments that was done for:

- Generating and evaluating embeddings for an English dataset.
- Align pretrained embeddings of English and Hindi languages and test them.
- Some miscellaneous on bonus tasks

2 Objective

Explore methods for generating dense word embeddings from a corpus and assess their quality through various evaluations.

3 Dataset Used

The dataset employed is a high-quality English corpus with predefined train and test splits. The train split contains approximately 153k sentences, while the test split consists of about 170k sentences. For the purposes of our task, we combine these and sample a total of 300k sentences. These sentences are then used as the foundation for generating dense word embeddings and for subsequent evaluation experiments.

4 Methodology

4.1 Generate Matrices:

- **Create cooccurance matrix:** To generate dense word embeddings, we first construct a co-occurrence matrix from the dataset. Each entry (i, j) in the matrix represents how often word i appears in the context of word j within a fixed context window. Since the matrix size grows quadratically with the vocabulary size, we restrict our vocabulary to frequently occurring words only. Specifically, we include words that occur with a frequency greater than or equal to 10. After applying this threshold, the resulting vocabulary size is 30,486. The vocabulary is sorted in descending order of frequency before constructing the matrix to prioritize high-frequency words.
- **Generating PPMI matrix:** After constructing the raw co-occurrence matrix, we transform it into a *Positive Pointwise Mutual Information (PPMI)* matrix. The PPMI value between two words w_i and w_j is defined as:

$$\text{PPMI}(w_i, w_j) = \max\left(0, \log \frac{P(w_i, w_j)}{P(w_i) \cdot P(w_j)}\right),$$

where $P(w_i, w_j)$ is the probability of words w_i and w_j co-occurring within the context window, and $P(w_i), P(w_j)$ are their marginal probabilities. This transformation is done to reduce the dominance of very frequent words and to highlight informative associations between words. Unlike the raw co-occurrence counts, the PPMI matrix captures meaningful statistical relationships that are more useful for downstream embedding generation.

- **Generating Shifted PPMI (SPPMI) matrix:** Following Levy and Goldberg [4], we extend the PPMI matrix by introducing a shift factor related to negative sampling. The Shifted PMI is defined as:

$$SPMI(w_i, w_j) = PMI(w_i, w_j) - \log k$$

where k is the number of negative samples. Similar to PPMI, we keep only the positive values to form the SPPMI matrix:

$$SPPMI(w_i, w_j) = \max\left(0, SPMI(w_i, w_j)\right).$$

In our experiments, we set $k = 5$.

- **Choosing window length:** We experiment with various context window lengths. Using smaller window sizes results in lower co-occurrence counts for less frequent words, which may negatively impact their ability to generate meaningful embeddings later on. On the other hand, larger window sizes increase co-occurrence counts but introduce noise, as words that are not semantically related may still appear within the same wide context. Balancing these trade-offs, we choose a window size of 5 for our experiments.

4.2 Dimension Reduction

- **Singular Value Decomposition (SVD):** To reduce the high-dimensional SPPMI matrix into compact dense embeddings, we apply truncated Singular Value Decomposition (SVD). The coocurance matrix is factorized as $COOC = U\Sigma V^\top$, and the top-d singular vectors from $U\Sigma^{1/2}$ are used as the final word embeddings. We experiment with different values of d and compute the cumulative explained variance to determine how much information is retained. By selecting d such that roughly 80–90% of the variance is captured, we obtain embeddings that are both informative and efficient for downstream tasks. This approach is inspired by Latent Semantic Analysis (LSA) [3] and is theoretically connected to neural embedding methods [4].

- **PCA:** Principal Component Analysis (PCA) is another linear dimensionality reduction technique applied to the cooccurance matrix. PCA identifies orthogonal directions (principal components) that capture the maximum variance in the data. By projecting the high-dimensional cooccurance matrix onto the top-d principal components, we obtain dense word embeddings of shape (V, d), where V is the vocabulary size. To select an appropriate number of dimensions d, we compute the cumulative explained variance:

$$\text{Cumulative variance}(d) = \frac{\sum_{i=1}^{d} \lambda_i}{\sum_{i=1}^{n} \lambda_i},$$

where λ_i is the eigenvalue corresponding to the i-th principal component. Dimensions are chosen to capture roughly 80–90% of the total variance. The resulting embeddings are evaluated intrinsically (word similarity, word analogy) and extrinsically (downstream NLP tasks) to ensure quality.

- **Non-negative Matrix Factorization (NMF):** We also experiment with Non-negative Matrix Factorization to obtain word embeddings. The non-negative coocurance matrix X is factorized as $X \approx WH$, where W provides the word embeddings and H represents the

latent basis for reconstruction. The non-negativity constraint ensures that embeddings are parts-based and interpretable. We vary the number of latent dimensions d and select the value that balances reconstruction error and embedding compactness. Embeddings are evaluated both intrinsically (word similarity) and extrinsically (downstream NLP tasks) to verify their quality.

4.3 Choosing appropriate value of D

We employ 2 methods here: SVD and PCA. After that, we choose the appropriate value of d corresponding to many values of D. We make some number of runs from 100 to 500, and we choose the value of D which covers about 90% of the explained variance. For out experiments we choose a D value of 300. We also have to make a tradeoff as larger value of D could result in more computation in other downstream tasks.

4.4 Word Similarity Evaluation

We evaluated our embeddings on three standard benchmarks: **WordSim-353** (353 word pairs with human similarity ratings), **MEN** (3,000 pairs focusing on general relatedness), and **SimLex-999** (999 pairs designed to capture strict similarity rather than association). For each dataset, we computed cosine similarities between word pairs (when both words were present in the vocabulary) and measured alignment with human judgments using Spearman's rank correlation.

In our case, the results for WordSim-353 and MEN were `NaN`, while SimLex-999 yielded a very low correlation. The `NaN` values occur because no dataset pairs overlapped with our vocabulary, reflecting the domain-specific nature of our embeddings and the consequent mismatch with these general-purpose benchmarks.

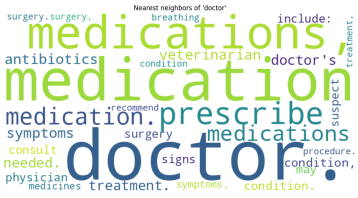

Figure 1: 2D point cloud visualization of a subset of embeddings.

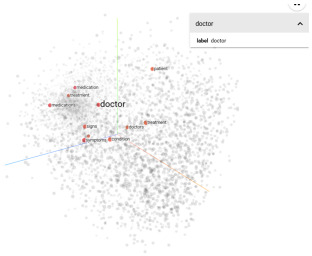

Figure 2: Snapshot of the TensorBoard Embedding Projector interface.

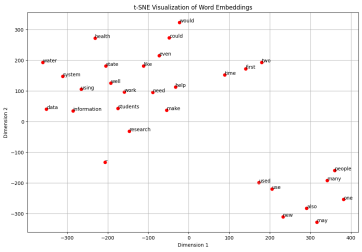

Figure 3: Images of some words together after doing TSNE.

4.5 Embedding Visualization

To qualitatively inspect the learned embeddings, we employed two complementary visualization techniques. First, we projected a subset of word vectors into two dimensions using dimensionality reduction methods such as PCA and t-SNE. The resulting scatter plots (point clouds) allow us to observe semantic groupings, where related words (e.g., *doctor*, *nurse*, *patient*) form localized clusters distinct from unrelated terms (e.g., *computer*, *network*, *data*). Second, we loaded the full embedding space into the **TensorBoard Embedding Projector** [1], which supports PCA, t-SNE, and UMAP visualizations along with nearest-neighbor queries for any word. We also do TSNE of some sample words to visualise the results in Figure 3.

4.6 Word2Vec Embedding Evaluation

We evaluated pre-trained Word2Vec [5] embeddings (Google News, 300-dimensional) on three standard word similarity benchmarks: **WordSim-353**, **MEN**, and **SimLex-999**. For each dataset, we computed the cosine similarity between word pairs (after lowercasing and filtering out missing words) and compared these predictions against human-annotated similarity scores using Spearman's rank correlation.

During preprocessing, some entries in the datasets were missing or non-string values, which were safely ignored. Word pairs where either word was not present in the Word2Vec vocabulary were skipped. As a result, for WordSim-353 and MEN, none of the word pairs overlapped with the vocabulary after filtering, yielding `NaN` Spearman correlations. For SimLex-999, a reasonable overlap existed, producing a Spearman correlation of approximately 0.40. This demonstrates that Word2Vec embeddings capture semantic similarity to a moderate extent for words present in the vocabulary, while coverage limitations affect performance on certain benchmarks.

4.7 Results

From the above evaluation, we can see that both WordSim-353 and MEN may not be such good evaluation tools, even for SimLex, the correlation values are coming pretty less. For neural based method, word2vec, shows higher correlation than fixed window sized cooccurance matrix.

5 Cross-Lingual Embedding Alignment

5.1 Datasets:

We use pre-trained embeddings for English and Hindi obtained from the FastText wiki vectors: `wiki.en.vec` for English and `wiki.hi.vec` for Hindi. These embeddings provide high-quality vector representations for words in both languages and serve as the basis for alignment.

5.2 Procrustes Alignment:

To align English and Hindi embeddings, we learn a linear mapping W that minimizes the Frobenius norm between the transformed source embeddings XW and the target embeddings Y:

$$W^* = \arg\min_{W \in \mathbb{R}^{d \times d}} ||XW - Y||_F$$

Optionally, W can be constrained to be orthogonal to preserve distances and angles. The optimal solution is computed using Singular Value Decomposition (SVD) as:

$$U\Sigma V^\top = \text{SVD}(Y^\top X), \quad W^* = UV^\top$$

After alignment, the embeddings can be evaluated via bilingual dictionary induction or cross-lingual word similarity tasks. This method preserves geometric structure and provides an efficient, strong baseline for supervised cross-lingual embedding alignment. We can see some of the results as follows in Table 1. For evaluation, we calculate Top-k nearest values and calculate precision@1 and precision@5.

5.3 Linear Alignment:

In addition to Procrustes alignment, we also experiment with a simple linear mapping that minimizes the squared reconstruction error between English and Hindi embeddings. Given source embeddings X (English) and target embeddings Y (Hindi), we learn a linear transformation W such that:

$$W^* = \arg\min_W ||XW - Y||_F^2$$

where $|| \cdot ||_F$ denotes the Frobenius norm. Optionally, small regularization can be added to improve numerical stability:

$$W^* = \arg\min_W ||XW - Y||_F^2 + \lambda ||W||_F^2$$

After learning W, the English embeddings are mapped as $X_{\text{mapped}} = XW$, and translations are retrieved by finding the nearest Hindi embeddings using cosine similarity. This approach provides a straightforward baseline and allows direct comparison with Procrustes alignment.

English Word	Hindi (Procrustes)	Score	Hindi (Linear)	Score	Hindi (CCA)	Score
dog	कुत्ते	0.6644	कुत्ते	0.7432	कुत्ते	0.8509
	जानवर	0.5866	जानवर	0.6670	जानवर	0.6706
	जानवरों	0.5026	जानवरों	0.6154	जानवरों	0.6397
	पशु	0.4985	पशु	0.5818	भालू	0.6203
	जंगली	0.4882	भालू	0.5813	पशु	0.6120
house	हाउस	0.6089	घर	0.6732	हाउस	0.8051
	मकान	0.5561	हाउस	0.6590	घर	0.7183
	घर	0.5335	मकान	0.6545	मकान	0.6777
	भवन	0.5015	कमरे	0.5907	घरों	0.6143
	कमरे	0.4690	भवन	0.5729	कमरे	0.5566
water	पानी	0.7322	पानी	0.8006	पानी	0.8924
	जल	0.7202	जल	0.7565	जल	0.8911
	गर्म	0.5184	गर्म	0.6208	सिंचाई	0.6623
	सिंचाई	0.5167	सिंचाई	0.6110	गर्म	0.6494
	पीने	0.4956	हवा	0.5678	नदियों	0.5652
school	स्कूल	0.7971	स्कूल	0.8344	स्कूल	0.9416
	विद्यालय	0.7270	विद्यालय	0.7475	विद्यालय	0.8810
	स्कूलों	0.6504	स्कूलों	0.7389	स्कूलों	0.8257
	कॉलेज	0.6167	कॉलेज	0.7080	महाविद्यालय	0.7198
	महाविद्यालय	0.6139	महाविद्यालय	0.6943	माध्यमिक	0.7151

Table 1: Comparison of top-5 Hindi translations for selected English words using Procrustes, Linear, and CCA alignment. Scores indicate cosine similarity.

5.4 Canonical Correlation Analysis (CCA) for Alignment

Canonical Correlation Analysis (CCA) provides an alternative method for aligning bilingual word embeddings. Unlike Procrustes alignment, which learn a single mapping from one language space to another, CCA simultaneously learns linear projections for both languages into a shared latent space.

Formally, let $X \in \mathbb{R}^{n \times d}$ denote the matrix of English embeddings and $Y \in \mathbb{R}^{n \times d}$ the matrix of Hindi embeddings for n seed dictionary pairs, each of dimensionality d. CCA seeks projection matrices $W_x, W_y \in \mathbb{R}^{d \times k}$ such that:

$$\max_{W_x, W_y} \text{corr}(XW_x, YW_y),$$

where $\text{corr}(\cdot, \cdot)$ measures the correlation between the projected embeddings. This ensures that the projected English and Hindi embeddings are maximally correlated along each of the k canonical dimensions.

The advantage of this approach is that it constructs a *symmetric shared space*, where both English and Hindi embeddings are transformed, rather than forcing one space to match the geometry of the other. This makes CCA particularly effective when the two embedding spaces have different distributions or anisotropic structures. After projection, translation equivalents are placed close together, enabling cross-lingual retrieval via nearest-neighbor search in the common space.

We have evaluated the results on our seed dictionary and evaluated the precision here.

Alignment Method	**P@1**	**P@5**
Linear Mapping	0.6013	0.8824
Procrustes	0.6100	0.8950
CCA	0.6122	0.9185

Table 2: Comparison of top-k precision for Linear Mapping and Procrustes alignment and CCA mapping.

6 Bonus Task- Harmful Associations

6.1 Identification and Evaluation of harmful associations:

As fixed window embeddings contain harmful associations, we want to find ways to mitigate that. The first step to doing that would be to identify harmful associations, like gender bias, racial bias, religious bias and so on. One way to evalaute this would be create sentences or situations where ”neutral” associations should be used but arent, and evaluate out embeddings based on a score.

Given a target word w and attribute sets A and B, we compute its differential association:

$$s(w, A, B) = \frac{1}{|A|} \sum_{a \in A} \cos(\vec{w}, \vec{a}) - \frac{1}{|B|} \sum_{b \in B} \cos(\vec{w}, \vec{b}),$$

This gives us Cohen’s score and a higher d [2] indicates stronger bias.

6.2 Experiment: Bias Evaluation in BERT using WinoBias

We do an experiement using BERT model on the WinoBias [6] which will attempt to find the gender bias in the BERT model using the WinoBias dataset.

6.3 Dataset

We use the WinoBias dataset [6], a benchmark designed to measure gender bias in coreference resolution systems. The dataset contains sentences with occupations and pronouns in two conditions:

- **Pro-stereotyped**: the pronoun aligns with societal gender stereotypes (e.g., "The developer argued with the designer because **he** was not cooperative.").
- **Anti-stereotyped**: the pronoun does not align with stereotypes (e.g., "The developer argued with the designer because **she** was not cooperative.").

For evaluation, the target pronoun is masked, and the model must predict the correct pronoun given the context.

6.4 Methodology

We fine-tuned evaluation on a pretrained `bert-base-uncased` model. Each sentence was tokenized and the target pronoun replaced by the `[MASK]` token. For prediction, we extracted probabilities over the candidate pronouns {`he, she, him, her`}. A prediction was considered correct if the highest-probability pronoun matched the gold label.

We compute:

- Accuracy on **pro-stereotyped** and **anti-stereotyped** subsets.
- The **gap** between these accuracies, where a large gap indicates bias.
- **Cohen's** d, an effect size measure of bias, calculated from the distributions of model confidence scores.

6.5 Results

	Pro-stereotyped	Anti-stereotyped	Gap
Accuracy	0.549	0.520	0.029
Cohen's d		0.029	

Table 3: Performance of BERT on WinoBias.

6.6 Discussion

The results suggest that BERT achieves similar performance on pro- and anti-stereotyped cases, with a small but measurable bias gap of 2.9%. The low Cohen's d (0.029) indicates a very weak effect size. While bias is present, the magnitude here is smaller than typically observed in earlier contextual models, highlighting the need for more sensitive and large-scale evaluations.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Jacob Cohen. *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Erlbaum Associates, 1988.

[3] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

[4] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, volume 27, pages 2177–2185, 2014.

[5] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[6] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. Gender bias in coreference resolution: Evaluation and debiasing methods. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 15–20, 2018.