Language Representation

Tasks Completeted

1. Part 1 - Dense Representations
    - Constructed co-occurence matrix - determined the most appropriate window size
    - Dimensionality reduction - computed the most suitable dimension in reduced space
    - Evaluate the quality of embeddings - Evaluated embeddings across 3 tasks/methods: Word similarity, Analogy reasoning, Clustering and Visualization
    - Neural Embeddings - Evaluated Glove embeddings using same methodology as before and compared the results with that of co-occurrence based embeddings
2. Part 2 - Cross-Lingual Alignment
    - Used pre-trained fasttext embeddings for both Hindi and English.
    - Used Procrustes alignment to learn transformation for projecting Hindi embeddings and English embedding space
    - Evaluated the alignment quantitatively using MUSE dataset. Also used a self-generated dataset for this evaluation
3. Part 3 - Harmful Associations (Bonus)
    - Analysed harmful associations in both static and contextual embeddings.
    - Used Glove 300 dimensional embeddings for static analysis. Quantitative analysis was done using the WEAT test.
    - Used Bert-base-uncased for contextual analysis and qualified my analysis using Causal Language Modeling. Generated sentence likelihood scores for different sentence to assess the stereotype present in contextual embeddings

Note - I explored different methodologies for both Part 2 and Part 3 but due to compulational and time constraints, stuck to the above listed implementations in my code and analysis.

Part 1: Dense Representations

1. Co-Occurrence Matrix Construction

Methodology

Data Preprocessing

Before constructing the co-occurrence matrix, the raw text corpus was preprocessed with the following steps:

1. **Lowercasing:** Converted all text to lowercase to maintain consistency.
2. **Removing Punctuation and Special Characters:** Eliminated non-alphanumeric characters that do not contribute meaningful semantic information.
3. **Removing Extra Spaces:** Cleaned up excessive spaces to standardize text formatting.
4. **Stopword Removal:** Removed common stopwords that would otherwise dominate the matrix and obscure meaningful relationships.

These preprocessing steps helped reduce noise, improve computational efficiency, and focus on semantically relevant word associations.

Construction of Co-Occurrence Matrix

A co-occurrence matrix was built where each row and column represented a word in the vocabulary. Each cell in the matrix contained the frequency with which the word in the row appeared in the context of the word in the column within a given window size.

- The vocabulary size was **162,625 words**, leading to a **162,625 x 162,625** matrix.
- Multiple matrices were generated for different window sizes (2-10) to analyze the effect of context size on word relationships.

Selecting the Optimal Window Size

Three evaluation techniques were applied to determine the best window size:

1. **Singular Value Decay Rate**

   Singular Value Decomposition (SVD) (SVD — Single Value Decomposition) was performed on the co-occurrence matrices, reducing them to 50 dimensions. The co-occurrence matrix is high-dimensional, and SVD helps reduce its complexity while retaining meaningful word associations. The rate at which singular values decay gives insights into how much information is concentrated in the leading components.

- Larger singular values suggest important information about word relationships.
- A **faster decay** means that the first few singular values capture most of the information, and the remaining ones contribute very little. This often suggests that the word relationships are relatively simple or that the matrix is dominated by a few strong patterns.
- A **slower decay** means that the information is more evenly distributed across the components. This suggests a more complex structure of word relationships, with many factors contributing to the co-occurrence patterns.

2. **Jensen-Shannon Distance**

   This method measures the similarity between probability distributions of consecutive matrices. If the JS distance is low (KL Divergence for Machine Learning), it means the matrices are nearly identical, indicating diminishing returns from increasing the window size further.

   - To measure the similarity between matrices of consecutive window sizes, the Jensen-Shannon (JS) distance was calculated.
   - A low JS distance implies minimal variation between matrices.

3. **Entropy Calculation**

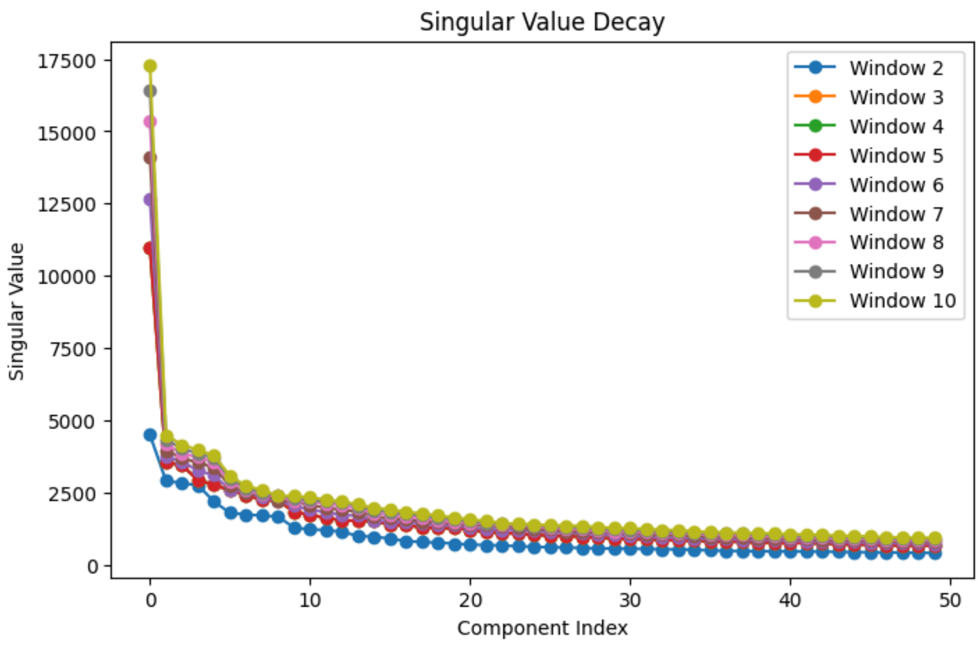
   Entropy measures the amount of disorder or uncertainty in a system. A lower entropy value means a matrix contains more structured, informative content rather than noise.

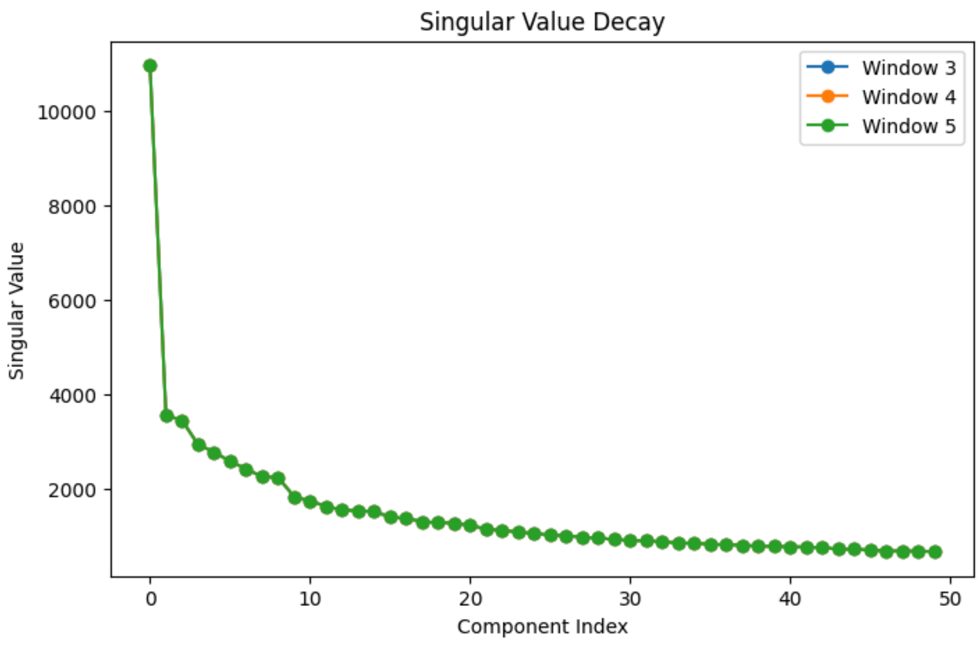
   - Entropy was computed for each matrix, where lower entropy values indicate higher information density.

Results and Findings

1. **Singular Value Decay rate**

   From the below 2 graphs, it can be observed that as window size increases, the drop between 1st and 2nd singular value also increases. Also, the first singular value exhibited a significant increase when moving from window size **2 to 3**, remained stable for **4 and 5**, and then showed smaller increases for larger window sizes. This suggests that **window sizes 3-5 provide the best balance between capturing high information content and maintaining structural consistency.**





2. **Jensen-Shannon Distance**

Jensen-Shanon distance between window 2 and 3: 0.01681496934621766
Jensen-Shanon distance between window 3 and 4: 0.01681496934621766
Jensen-Shanon distance between window 4 and 5: 0.01681496934621766
Jensen-Shanon distance between window 5 and 6: 0.019849864509691003
Jensen-Shanon distance between window 6 and 7: 0.022540440728076334
Jensen-Shanon distance between window 7 and 8: 0.024963528553191883
Jensen-Shanon distance between window 8 and 9: 0.027169245597886194

Jensen-Shanon distance between window 9 and 10: 0.02916597967858617

The JS distance between all the window sizes was very low, showing that the information contained in the matrices did not change very much between the window sizes. Hence, **increasing the window size does not lead to any significant change in representation.**

3. **Entropy**

   The entropy of all the matrices was very similar with **window sizes 3, 4 and 5 having the lowest entropy**, meaning that they have the most useful information.

Final Selection: Window Size 5

Based on these evaluations, **window size 5** was chosen as the optimal setting because:

- It provided a good balance in singular value decay, maintaining meaningful word associations.
- JS distance analysis showed minimal variation, indicating that word relationships remained stable beyond this point.
- Entropy remained low, confirming that the co-occurrence matrix retained valuable information.
- It aligns with standard NLP practices, where a window size of **5** is commonly used as it is neither too narrow (overly restrictive) nor too wide (introducing spurious relationships).

Conclusion

A co-occurrence matrix was successfully constructed from a preprocessed English text corpus. Through systematic evaluation using eigenvalue decay, Jensen-Shannon distance, and entropy calculation, window size **5** was determined to be the most effective choice. This analysis ensures that the word embeddings generated from this matrix will capture meaningful word relationships while maintaining computational efficiency.

2. Dimensionality Reduction

Methodology

Objective

The original co-occurrence matrix is extremely large (162,625 x 162,625), making computations expensive and memory-intensive. To efficiently store and process word relationships, dimensionality reduction techniques were applied to transform the high-dimensional matrix into a lower-dimensional representation while retaining meaningful word associations.

Technique Used: Truncated Singular Value Decomposition (Truncated SVD)

Truncated SVD was chosen as the primary dimensionality reduction method due to the following advantages:

- **Computational Efficiency:** Unlike full SVD, which computes all singular values, Truncated SVD (Truncated SVD and its Applications, TruncatedSVD — scikit-learn 1.6.1 documentation, Truncated SVD for Dimensionality Reduction in Sparse Feature Matrices | by Rukshan Pramoditha) retains only the top *k* singular values and vectors, significantly reducing computational complexity.
- **Preserves Semantic Structure:** The most significant singular values capture the core semantic relationships between words, ensuring that essential linguistic patterns are retained while discarding noise.
- **Handles Sparse Data Well:** The co-occurrence matrix is large and sparse, making Truncated SVD a more practical choice than PCA, which requires centering the data and is computationally more expensive.

Choosing the Optimal Number of Dimensions (d)

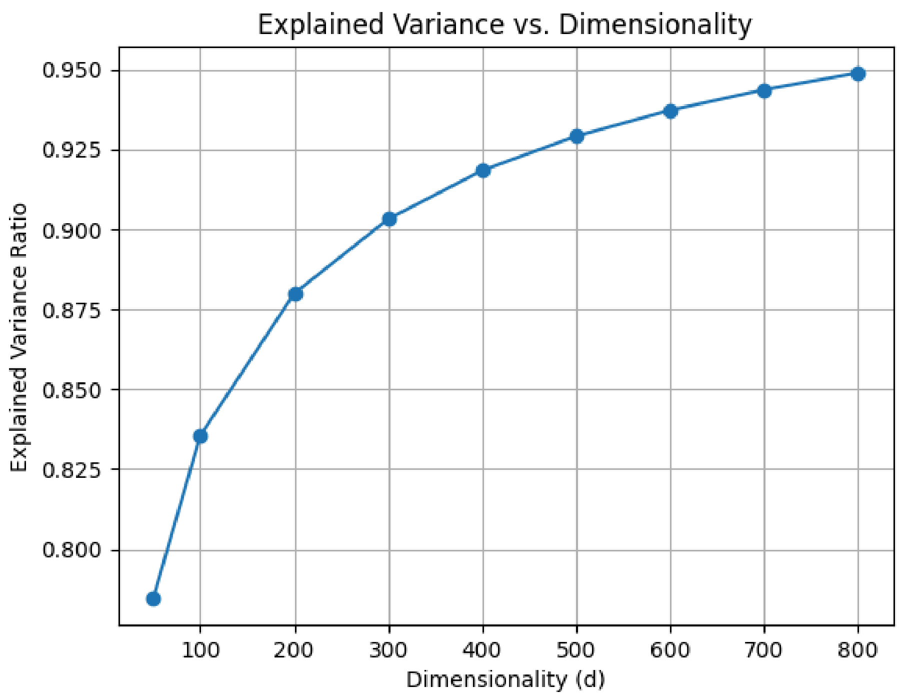
Selecting an appropriate number of dimensions (*d*) is crucial to balance information retention and computational efficiency. Two methods were used to determine the optimal value of *d*:

1. **Elbow Method (Explained Variance Analysis)**
    - The singular values of the matrix represent the amount of variance captured by each dimension.
    - The explained variance ratio was plotted against different values of *d* to identify the "elbow point," where additional dimensions contribute diminishing returns in information gain.
    - The goal was to find the smallest *d* that retains the majority of useful information.
2. **Word Analogy Evaluation (King - Man + Queen ≈ Woman Test)**
    - Word embeddings should capture semantic relationships effectively.

- The famous analogy test **(King - Man + Woman ≈ Queen)** was used to assess how well different dimensionalities preserve meaningful relationships.
- The cosine similarity between the resulting vector and the expected word (Queen) was computed for different values of *d*.
- Higher cosine similarity indicates that the word embeddings effectively capture word relationships.

Results and Findings

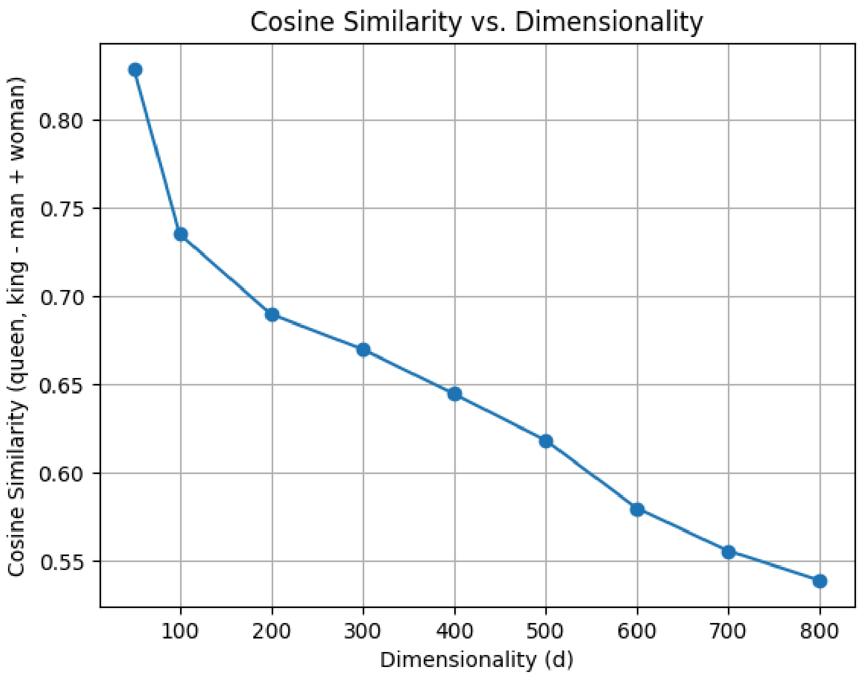
1. Elbow Method (Explained Variance Analysis)



Observation:

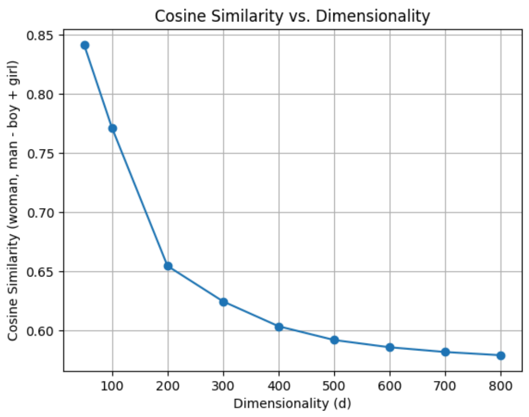
- A sharp increase in variance was observed from **50 to 200 dimensions** (~0.775 → ~0.875).
- Beyond **200 dimensions**, the increase in variance became marginal.
- The **elbow point** was identified at **200 dimensions**, indicating that increasing beyond this value does not significantly improve the captured information.

2. Word Analogy Evaluation (Cosine Similarity for "King - Man + Woman ≈ Queen")



Observation:

- **50 dimensions achieved the highest cosine similarity (0.78)**, meaning it preserved word relationships best.
- As dimensionality increased, cosine similarity **decreased**, suggesting that **higher dimensions diluted the strong semantic relationships** rather than improving them.
- This indicates that a **lower-dimensional representation may be better at capturing meaningful word relationships** in analogy-based tasks.
- A similar was also observed for the analogy "Man - Boy + Girl ≈ Woman" which supports the above observation



Final Selection of Dimensionality

Given that the two evaluation methods produced different results, both **50 and 200 dimensions** were selected for further testing in downstream tasks:

- **50 Dimensions**: Optimal for tasks requiring strong semantic relationships (e.g., analogy tasks).
- **200 Dimensions**: Provides a good balance, retaining most of the variance while keeping embeddings compact.

Conclusion

Dimensionality reduction was successfully applied to the co-occurrence matrix using **Truncated SVD**, significantly reducing storage and computational costs while preserving meaningful word associations. Through a **variance analysis (Elbow Method)** and a **word analogy test**, it was found that:

- **200 dimensions** provided the best balance in retaining variance while keeping the embeddings compact.
- **50 dimensions** performed best in capturing word analogies, making it useful for certain NLP tasks.
- Both **50 and 200 dimensions** were retained for further evaluation in downstream applications.

3. Evaluating Word Embeddings

Methodology

After constructing word embeddings with 50 and 200 dimensions, their quality was evaluated using three different approaches: word similarity analysis, analogy tests, and clustering/visualization. Additionally, both raw and normalized embeddings were assessed to determine the effect of normalization on performance.

Why Normalize?

Normalization ensures that embeddings focus on direction rather than magnitude, improving various distance-based evaluations:

- **Focus on Semantic Similarity:** Cosine similarity, commonly used for word similarity tasks, depends only on the angle between vectors, not their magnitudes. Normalizing vectors to unit length ensures that comparisons are based solely on directional alignment.
- **Improved Consistency and Stability:** Normalization reduces variance in magnitudes, leading to more stable evaluation results, regardless of embedding dimensions or training variations.
- **Better Clustering Performance:** In clustering algorithms like k-means, unnormalized embeddings may form clusters dominated by magnitude differences rather than semantic meaning, while normalization aligns clusters more meaningfully.

1. Word Similarity Evaluation

Word similarity tests measure how well embeddings align with human perception of word relationships. Three widely-used datasets were employed:

- **SimLex-999** (Hill et al., 2015): Focuses on strict similarity rather than relatedness, making it more challenging for embeddings to score well.
- **WordSim-353** (Finkelstein et al., 2002): Contains word pairs with human similarity ratings but blends both relatedness and similarity.
- **MEN** (Bruni et al., 2014): Captures human judgments for visual and linguistic associations, offering a diverse test set.

For each dataset, the cosine similarity of word pairs was computed, and the Pearson correlation coefficient between cosine similarities and human ratings was calculated. Higher correlation values indicate that the embeddings accurately reflect human-perceived word relationships.

Results:

Dataset	Correlation (d=50)	Correlation (d=200)	Correlation (d=50, Norm)	Correlation (d=200, Norm)
SimLex-999	0.1344	0.1375	0.1344	0.1375
WordSim-353	0.3675	0.3710	0.3675	0.3710
MEN	0.3737	0.4129	0.3737	0.4129

Findings:

- Normalization had no effect on word similarity results, as expected, since cosine similarity is already independent of vector magnitude.

- SimLex-999 had the lowest correlation due to its emphasis on similarity rather than relatedness, which is difficult for embeddings trained on co-occurrence.
- WordSim-353 and MEN showed reasonable correlation, with MEN achieving the highest values.
- Increasing dimensionality from 50 to 200 resulted in slight improvements, but the gains were not substantial.

2. Analogy Evaluation

Analogies test whether embeddings capture structured relationships between words. The **Bigger Analogy Test Set (BATS)** (Gladkova et al., 2016) was used, containing 40 analogy categories across four relationship types: morphology, lexicography, encyclopedic, and syntax-based analogies. Each test follows the form **w1:w2 = w3:w4**, where embeddings should correctly infer w4 using vector arithmetic: **cosine(em4, em2 - em1 + em3).**

Correctness was determined using a threshold where cosine similarity ≥ 0.5 was considered a correct prediction.

Results:

Model	**Accuracy (Correctness > 0.5)**	**Cosine Similarity**
d=50	0.5645	0.4152
d=200	0.4526	0.3573
d=50 (Norm)	0.6263	0.5457
d=200 (Norm)	0.3882	0.4136

Findings:

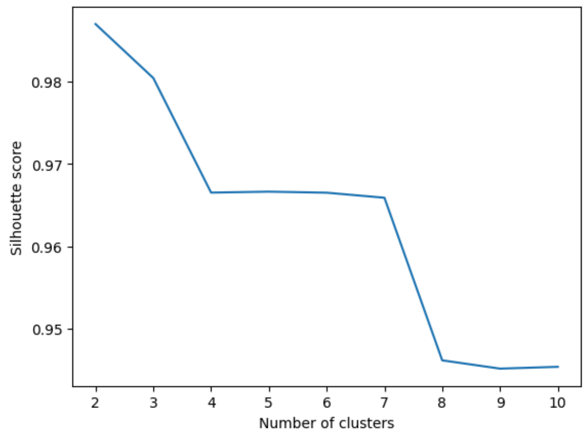
- **50-dimensional embeddings outperformed 200-dimensional embeddings** in analogy tasks, suggesting that lower-dimensional embeddings might generalize better for capturing structured relationships.
- **Normalization significantly improved performance in 50-dimensional embeddings**, confirming that removing magnitude effects helps with vector algebra.
- **200-dimensional embeddings performed worse when normalized**, likely due to increased sparsity, making it harder to capture fine-grained relationships.

3. Clustering and Visualization

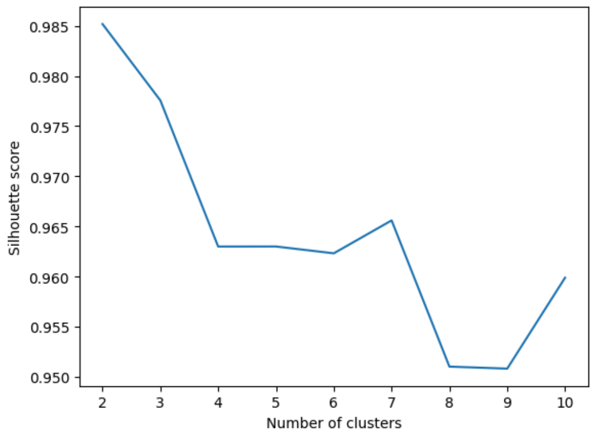
Clustering and visualization techniques were applied to analyze the structure of embeddings.

K-Means Clustering:

- A range of clusters (2-10) was evaluated using silhouette scores to determine the best clustering structure.
- Non-normalized embeddings had extremely high silhouette scores (>0.95), but this was due to magnitude-driven clustering rather than meaningful semantic grouping.

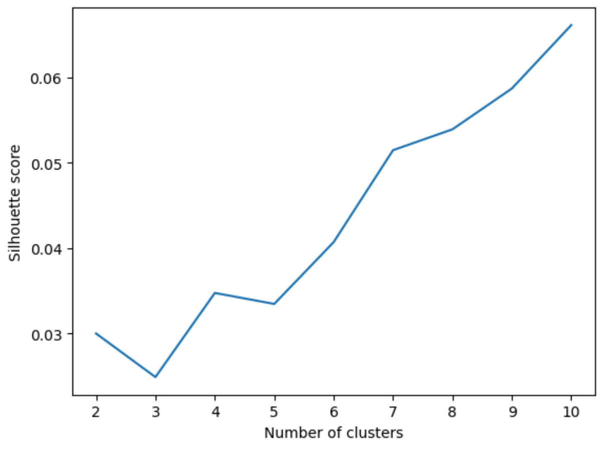


50 Dimensional embeddings

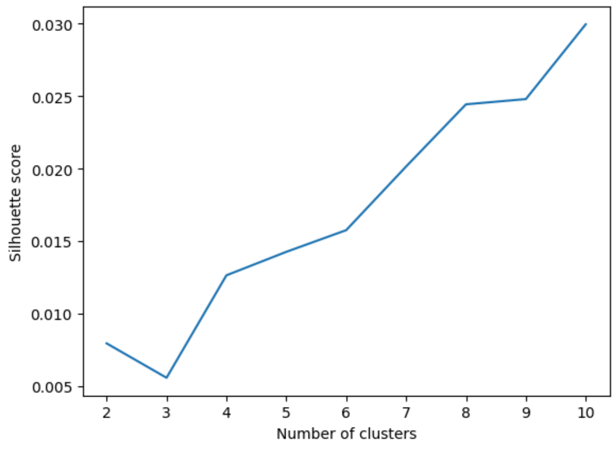


200 dimensional embeddings

- Normalized embeddings had very low silhouette scores (<0.03), indicating highly overlapping clusters and weak separation.



50 dimensional embeddings



200 dimensional embeddings

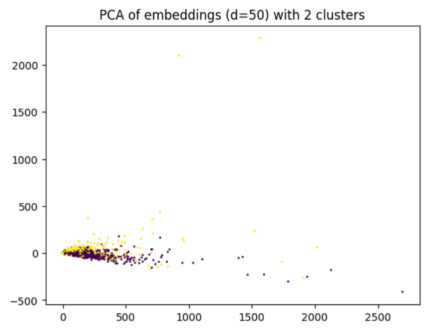
t-SNE and PCA Visualization:

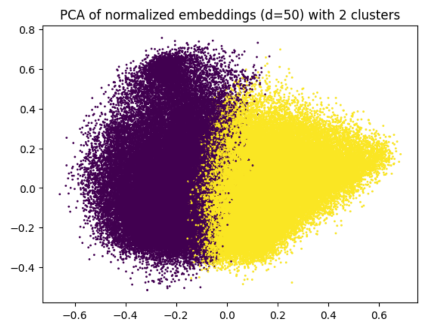
- Both techniques were applied to reduce embeddings to 2D for visualization.
- **Neither method produced well-separated clusters**, confirming that word embeddings did not form distinct, interpretable clusters in this representation.
- **50-dimensional embeddings provided slightly better structure than 200-dimensional embeddings, but overall separation remained poor.**

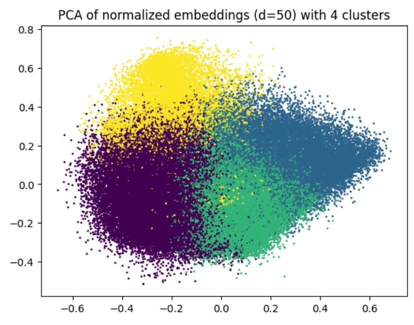
Final Selection: 50-Dimensional Normalized Embeddings

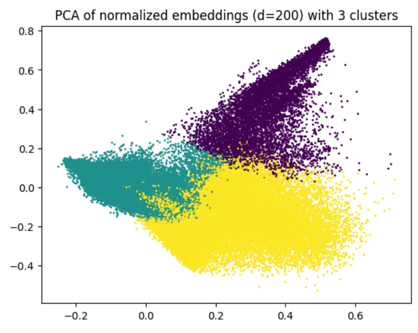
After evaluating all aspects, **50-dimensional normalized embeddings** were chosen as the best configuration:

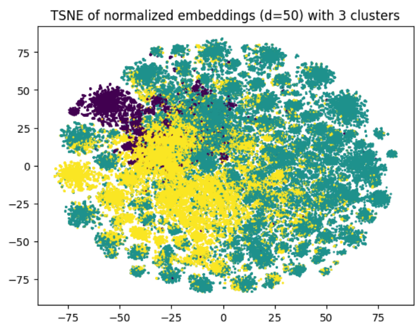
- **Performed best in analogy tasks**, showing strong structural relationships.
- **Performed reasonably well in word similarity tasks**, though dimensionality did not significantly affect results.
- Normalization improved analogy performance while having no negative effect on similarity evaluation.
- Higher-dimensional embeddings (200d) did not provide significant improvements and sometimes performed worse.
- Find some PCA and T-SNE plots below. The rest of the results are available in the Python notebook. These graphs show that none of the permutations result in good quality clustering.

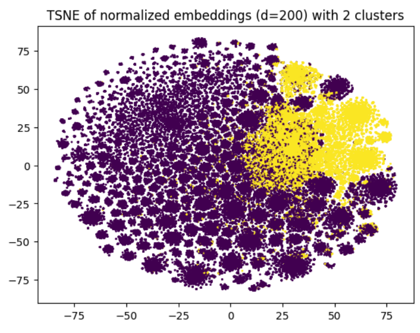












Conclusion

Three evaluation methods—word similarity, analogies, and clustering—were applied to assess embedding quality. Word similarity scores aligned reasonably with human judgment; analogy tasks highlighted the benefits of lower-dimensional embeddings with normalization, and clustering showed that embeddings lacked strong discrete cluster structures. Based on these findings, **50-dimensional normalized embeddings were selected as the final model**, balancing efficiency and performance.

4. Neural Word Embeddings

Methodology

To compare co-occurrence-based embeddings with neural embeddings, I evaluated pre-trained GloVe embeddings (glove-wiki-gigaword-50) against the best-performing co-occurrence-based embeddings (50-dimensional normalized). The same evaluation tasks—word similarity, analogy reasoning, and clustering/visualization—were applied to ensure a fair comparison.

For a direct comparison, only words present in the co-occurrence-based embeddings were retained from the GloVe embeddings. This prevents introducing new words that were not part of the previous analysis, ensuring that any observed differences result from the embedding method rather than dataset variations.

As an initial quality check, I computed the cosine similarity for the analogy "king - man + woman = queen." GloVe produced a score of **0.93**, while the best co-occurrence-based embeddings achieved **0.84**, indicating that the neural embeddings capture semantic relationships more effectively.

Results and Findings

1. Word Similarity

- SimLex-999 Correlation: **0.2667** (higher than 0.1374 for co-occurrence)
- WordSim-353 Correlation: **0.4981** (higher than 0.3709 for co-occurrence)
- MEN Correlation: **0.6509** (higher than 0.4129 for co-occurrence)

GloVe embeddings significantly outperformed co-occurrence-based embeddings across all three datasets. This aligns with expectations since neural methods like GloVe optimize word representations to capture semantic similarity beyond simple co-occurrence patterns. Unlike raw frequency-based methods, GloVe learns global co-occurrence statistics, meaning that even words that do not directly co-occur can have meaningful relationships.

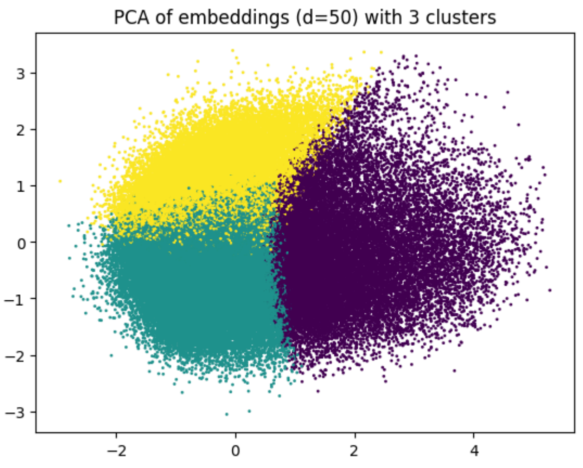
2. Analogy Reasoning

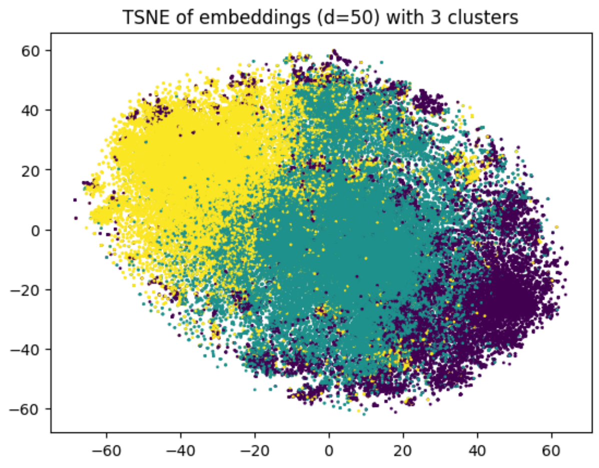
- Analogy Correctness: **0.6111** (compared to 0.6263 for co-occurrence)
- Analogy Cosine Similarity: **0.5509** (compared to 0.5457 for co-occurrence)

Interestingly, the co-occurrence-based embeddings achieved **slightly better correctness** than GloVe, even though the average cosine similarity was marginally higher for GloVe. This suggests that while neural embeddings capture more semantic meaning in general, the structure of relationships within the co-occurrence-based embeddings may be more rigidly aligned to simple arithmetic patterns. Since co-occurrence matrices capture direct count-based associations, they may excel in specific analogy cases where direct relationships dominate over abstract semantic similarity.

3. Clustering and Visualization

- Silhouette Scores: **0.02 - 0.08**, similar to co-occurrence-based embeddings
- PCA and t-SNE plots: Slightly better separation than co-occurrence embeddings but still lacking distinct clusters





Despite the advantages of neural embeddings in word similarity and analogies, their clustering behavior remains weak. This suggests that while GloVe embeddings form dense, contextually rich representations, they do not inherently create distinct clusters in a low-dimensional space. This is expected, as embeddings are designed for smooth, continuous relationships rather than discrete categories. The high-dimensional space they are originally trained in may encode semantic groupings more effectively than what is visible in reduced dimensions.

Conclusion

Neural embeddings (GloVe) consistently outperform co-occurrence-based embeddings in capturing semantic similarity and word relationships, as reflected in higher correlation scores across all word similarity datasets. Their effectiveness comes from learning global co-occurrence statistics rather than just local context windows, enabling them to model long-range dependencies and nuanced meanings. Overall, GloVe embeddings are superior for capturing rich word meanings and relationships, making them preferable for downstream NLP tasks requiring nuanced semantic understanding.

Part 2: Cross-Lingual Alignment

Methodology

To explore cross-lingual alignment, I used pre-trained monolingual word embeddings for English and Hindi from FastText: `cc.en.300.bin` and `cc.hi.300.bin`. FastText embeddings are particularly suited for this task because they capture subword information, which helps with morphologically rich languages like Hindi. Additionally, the authors of MUSE recommend using FastText embeddings for alignment with their dataset, making it a natural choice for this experiment.

Datasets

For learning a transformation between the embeddings, word pairs mapped to each other in both languages (i.e., exact translations) were required. Two datasets were used:

1. **MUSE (Multilingual Unsupervised and Supervised Embeddings)** (Zhao et al., 2017)**:** This dataset provides 8000 Hindi-English word pairs for training and 1960 for testing. It includes a diverse vocabulary with common words as well as more domain-specific terms.
2. **Self-Generated Dataset:** Since MUSE might not cover all necessary words, I created a supplementary dataset using ChatGPT. Generating this dataset required multiple iterations due to challenges like hallucinations, loss of context, and low generation per prompt. The final dataset contained 2400 word pairs, focusing on common words across topics like household items, science, arts, and sports. I used 2000 pairs for training and 400 for testing.

To assess initial similarity, I computed the average cosine similarity between corresponding word pairs in both test sets before alignment. The results were:

- MUSE Test Set: **0.011**

- Self-Generated Test Set: **0.013**

These near-zero values indicate that the English and Hindi embeddings reside in completely different spaces, confirming the need for alignment.

Alignment using Procrustes Analysis

To align Hindi embeddings with English embeddings, I applied **Procrustes Analysis (**3. Procrustes alignment — fmralign tutorials, Orthogonal Procrustes problem - Wikipedia**)**, which finds an optimal orthogonal transformation to align two vector spaces. This method is effective because:

- It preserves distances between word vectors, maintaining semantic relationships.
- It ensures an orthogonal transformation, preventing distortion of the original space.
- It has been widely used in prior work for cross-lingual alignment due to its efficiency and effectiveness.

The alignment process involved computing a rotation matrix using training word pairs:

1. Extract the embedding vectors corresponding to the **parallel word pairs**.
2. Solve for the **optimal orthogonal transformation matrix (R)** using the **Orthogonal Procrustes** method: $\mathbf{R = argmin_R || RX - Y||_F}$
   where **X** are the Hindi embeddings, **Y** are the English embeddings, and **R** is the transformation matrix.
3. Apply the learned transformation to the entire Hindi embedding space: **X' = XR.**

Results and Findings

After applying Procrustes alignment, I evaluated the transformation using both test sets:

Training Set Used	MUSE Test Cosine Similarity	Self-Generated Test Cosine Similarity
MUSE	0.38	0.46
Self-Generated	0.33	0.45

1. **Alignment works:** The increase from near **0.01 to 0.38-0.46** confirms that the transformation effectively brings Hindi embeddings closer to English embeddings.
2. **Comparable results across training sets:** Even though the self-generated dataset had only ¼ the number of training pairs compared to MUSE, it achieved similar alignment

quality. This suggests that a carefully curated set of common words might be sufficient for high-quality alignment.

3. **Higher scores for the self-generated test set:** This may be because the self-generated set contains more frequently occurring words, making it easier to align compared to MUSE, which includes less common words and stopwords that might introduce noise.
4. **Effectiveness of Procrustes Analysis:** The method successfully transformed Hindi embeddings into the English space while preserving word relationships, demonstrating its robustness.

Cross-lingual alignment using Procrustes analysis successfully mapped Hindi embeddings into the English embedding space, demonstrating a substantial improvement in cosine similarity. The results indicate that **training on common words alone can be as effective as using a large dataset**, reducing the need for extensive parallel corpora. Moreover, the self-generated test set consistently achieved better similarity scores, highlighting the importance of **choosing appropriate word pairs** for alignment. Future work can explore **non-linear alignment techniques** to further improve the robustness of cross-lingual embedding mappings.

Part 3: Bonus - Harmful Associations

Methodology

Pretrained word embeddings are trained on vast corpora, capturing semantic relationships between words. However, this also means that they can reflect and propagate biases present in the training data. This section evaluates gender bias in both static and contextual embeddings using well-established methodologies and datasets. The static embeddings are analyzed using the Word Embedding Association Test (WEAT), while the contextual embeddings are assessed using a causal language modeling approach with the StereoSet dataset.

1. Static Embeddings

For analyzing static embeddings, we use **GloVe embeddings (300 dimensions)** from `glove-wiki-gigaword-300`. The **Word Embedding Association Test (WEAT)** (Caliskan et al., 2017) is used as the evaluation metric, quantifying the association strength between different word categories.

WEAT Methodology: The WEAT score is computed as follows (Word Embedding Fairness Evaluation - KDnuggets):

1. **Target Sets:** Two sets of words, one corresponding to male-associated words and the other to female-associated words.

    - Male Words: ['man', 'male', 'he', 'brother', 'father', 'son', 'him', 'king', 'uncle', 'sir']
    - Female Words: ['woman', 'female', 'she', 'sister', 'mother', 'daughter', 'her', 'queen', 'aunt', 'madam']
2. **Attribute Sets:** Words that are typically associated with either male or female stereotypes:
    - Career vs. Family
    - Male vs. Female Professions
    - Male vs. Female Traits
    - Male vs. Female Hobbies
    - Pronouns (to validate test correctness)

3. **WEAT Calculation:** The association of a target word **w** with an attribute set **A** over another set **B** is given by:

   $$s(w, A, B) = \text{mean}_{a\in A} \cos(w, a) - \text{mean}_{b\in B} \cos(w, b) \quad (1)$$

   The final WEAT effect size is:

   $$d = \frac{\text{mean}_{x\in X} s(x,A,B) - \text{mean}_{y\in Y} s(y,A,B)}{\text{stddev}_{z\in X\cup Y} s(z,A,B)} \quad (2)$$

   A WEAT score close to **2** indicates strong bias in the expected direction because, according to the equation, a larger effect size occurs when the difference between male-associated and female-associated word similarities is significant, confirming the presence of expected stereotypes. Conversely, **-2** would indicate a reversal of the expected bias, and **0** means neutrality.

   For example, consider X is male target word set and Y is female target word set. Also A is stereotypical male and B is stereotypical female attributes. This means that s(x, A, B) is greater than s(x, A, B) as X is more closely related to A hence, first term of numerator in (2) is positive (from (1)). Similarly, second term of numerator of (2) should be negative or less positive than first term since Y is more closely related to B. So, if our bias assumption is correct, the numerator should turn out to be positive and negative of our assumption was opposite. It should be near 0 if the the attributes are neutral to both the targets.

2. Contextual Embeddings

For contextual embeddings, we analyze **BERT-base-uncased** using the **StereoSet dataset** (Nadeem et al., 2020), which consists of masked sentences with three variations:

- **Stereotypical**: Reinforces gender bias
- **Anti-stereotypical**: Opposes bias
- **Unrelated**: Nonsensical sentence (acts as a control)

Evaluation Metrics:

- **Sentence Likelihood**: Computed using causal language modeling, where the model predicts each token of the sentence instead of one masked token by passing same tokens in input_ids and labels. This helps us convert BERT's MLM for our CLM task to score likelihood of sentences.

  The score of a sentence is calculated using the normalized negative log-likelihood: P(sentence) = $e^{-\text{loss}}$ where **loss** is obtained from the BERT model's masked language modeling objective

- **Stereotype Score (SS):** Measures how often the model prefers the stereotypical over the anti-stereotypical sentence:

  $$SS = \frac{P(\text{stereotypical})}{P(\text{stereotypical})+P(\text{anti-stereotypical})} \times 100$$

  Ideal SS = 50 (no preference between stereotypical and anti-stereotypical choices).

- **Language Modeling Score (LMS):** Measures whether the model assigns higher likelihood to meaningful sentences:

  $$LMS = \frac{P(\text{stereotypical})+P(\text{anti-stereotypical})}{P(\text{stereotypical})+P(\text{anti-stereotypical})+P(\text{unrelated})} \times 100$$

  Higher LMS indicates a better understanding of meaningful sentences.

- **Idealized Score (IS):** Combines both SS and LMS to penalize biases and encourage meaningful predictions:

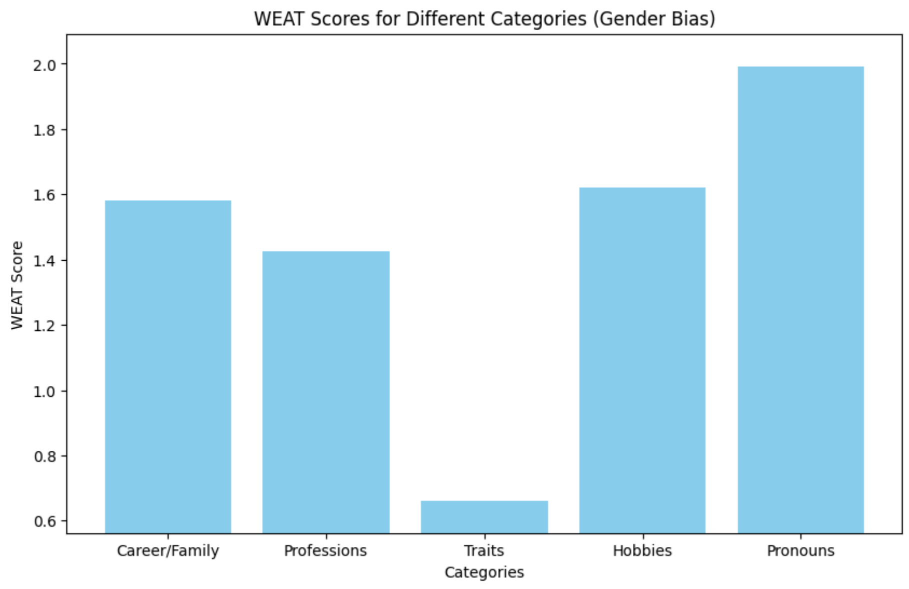
  $$IS = (100 - |SS - 50|) \times \frac{LMS}{100}$$

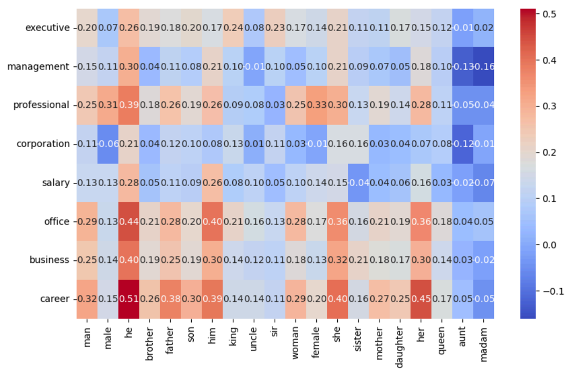
  Higher IS is desirable.

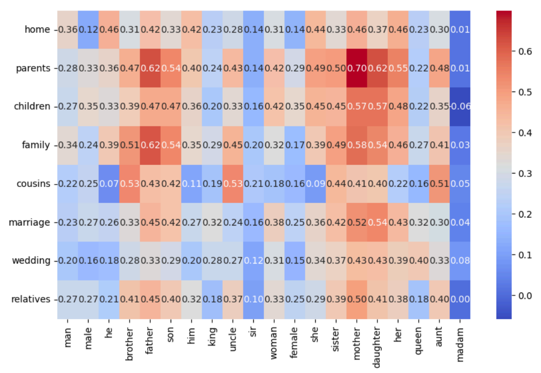
Results

1. Static Word Embeddings

Category	WEAT Score
Career vs. Family	1.5816
Male vs. Female Professions	1.4249
Male vs. Female Traits	0.6623
Male vs. Female Hobbies	1.6195
Pronouns (Control)	1.9918



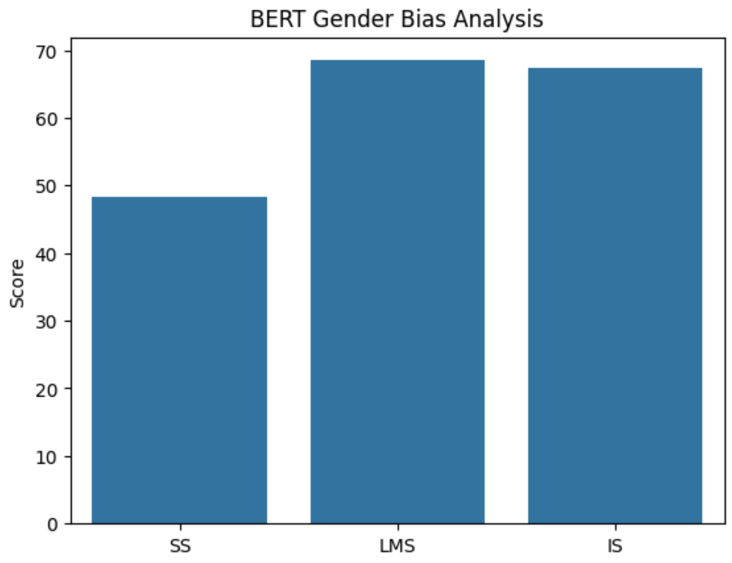




2. Contextual Word Embeddings

Metric	Score
SS	48.26

LMS	68.50
IS	67.31



Findings and Discussion

1. The analysis for static and contextual embeddings **differ in the way they are evaluated**. The static analysis uses WEAT scores. Since static embeddings do not change based on context, a **fixed similarity score using cosine similarity** is sufficient to evaluate inherent biases. This allows for direct comparisons between word pairs without requiring sentence-level evaluation.

   Unlike static embeddings, **contextual embeddings are dynamic**—the meaning of a word depends on the sentence in which it appears. Therefore, rather than comparing isolated word associations, BERT's performance is tested on **entire sentences**, where the model decides whether a stereotypical or anti-stereotypical completion fits better in context. We use CLM to evaluate this which assigns scores to the likelihood of a sentence.

2. **Static embeddings exhibit strong gender biases** due to their fixed nature and co-occurrence-based learning. This is evident from WEAT scores. The heatmaps attached also show this. In the left figure, when compared with career words, male target words show higher correlation and opposite can be seen in the right figure which is plotted against family words (other heatmaps present in ipynb files)
3. **Contextual embeddings show reduced bias**, as seen in the **SS score (~50)**, demonstrating that BERT does not inherently favor stereotypical statements.
4. **WEAT is suited for static embeddings, whereas causal language modeling is needed for contextual models**, as context determines bias in sentence generation.

5. **Static embeddings retain biases more strongly**, while **contextual embeddings adapt dynamically**, reducing bias but not eliminating it.

Conclusion

This analysis highlights the presence of gender bias in word embeddings and the importance of evaluation techniques for measuring harmful associations. While static embeddings like GloVe show **substantial bias**, contextual models like BERT mitigate bias to an extent but are **not completely bias-free**. Future work could explore **debiasing strategies**, such as **adversarial training** or **counterfactual data augmentation**, to reduce bias in both static and contextual representations.