Precog Report

Kabir Jamadar

1. Introduction

For this submission, I selected **Task 2: Language Representations** because it allowed me to engage with deeper questions around how language models understand and represent meaning. The task offered a compelling mix of algorithmic implementation and critical analysis, especially around evaluating alignment techniques and probing for social biases. This report walks through the methods I applied, decisions I made along the way, the results I observed, and key takeaways from the process.

2. Task completion

- While I have attempted all the tasks, there were a few experiments I couldn't get to due to time constraints.

3. Dense Representations

The goal of this task is to figure out how to turn a big collection of text into useful word representations. This is being done through co-occurrence based embeddings first, then checking how good they are with tests like word similarity, analogies and clustering. Later on, these embeddings are also compared with Word2Vec to see which works better.

3.1 Text Corpora, Preprocessing and Tokenization

For this task, I used a **300K sentence English news corpus from the Leipzig Corpora** collection.

After extracting the sentences, the following preprocessing was applied:

- **Lowercasing**: This ensures case consistency and avoids treating semantically identical words as different tokens, reducing unnecessary vocabulary size.

- **Removing URLs and Web Links**: After inspection, it was found that articles often contain hyperlinks that don't contribute to the semantic meaning of sentences. This also prevents noisy and irrelevant tokens from being included in the vocabulary, improving the overall vector quality.
- **Removing punctuation and Non-Alphabetic characters**: Punctuation and numeric characters often dont carry useful contextual meaning for word embedding models like co-occurrence matrices. Cleaning these elements helps focus only on content-bearing words, ensuring the co-occurrence matrix reflects true lexical relationship.
- **Tokenization**: Sentences need to be split into individual words to track co-occurrence counts and build the matrix. NLTK's tokenizer is fast and ideal for this task. Spacy or HuggingFace tokenizers often require loading large models, which is unnecessary overhead for this task.
- **Stopword Removal**: Words like "the", "and" appear too frequently but dont contribute meaningfully to semantic understanding. Removing them improves the clarity by focusing on content words and leads to a denser, more meaningful vector representations.
- **Lemmatization**: Words often appear in inflected forms (eg. run, running, ran). Lemmatization reduces these forms to a common base (eg run), helping the model to learn more robust and generalizable embeddings

To reduce noise and sparsity, I excluded words fewer than 5 times. Out of ~140K unique tokens, over 107K had frequency <5, contributing little statistical value. Lower thresholds (e.g., 1–4) retained too much noise, while higher ones (e.g., 6–10) risked discarding useful but moderately frequent content words. Setting min_count=5 gave me a cleaner vocabulary to work with.

3.2 Co-occurrence Matrix Creation

- **Choosing window sizes**:
    - To determine an appropriate context window size for matrix construction, I analyzed the distribution of tokenized sentence lengths from the corpus. The statistics revealed a mean sentence length of ~11 tokens, a median of 11, and a

maximum of 35 tokens. The distribution was right-skewed, with most sentences falling between 8 and 15 tokens.

    - Based on this: I chose to experiment with window sizes [5,8,10,12,18]
        - 5 captures tight, local context
        - 8-12 aligns with the statistics.
        - 18 serves as an upper bound, for long range semantic relationships.
- The co-occurrence matrix is highly sparse, out of millions of possible word pairs, only a small fraction-co-occur. Using numpy dense arrays would be computationally expensive and waste massive memory, since most entries would be 0.
- *lil_matrix* allows for efficient row-wise appending and modification, which matches the use case, updating pair counts as we scan through sentences. It's internal structure makes *A[i,j]+=1* fast when building the matrix incrementally.
- Once built, the matrix is needed for fast matrix operations (like svd) and efficient slicing during tasks. *Csr_matrix* is optimized for this with fast row access and optimized arithmetic, dot products and memory efficient storage.
- This split avoids performance bottlenecks and scales to large vocabularies cleanly.

3.3 Dimensionality Reduction

The original co-occurrence matrix is of shape N x N, where *N* is the vocabulary size (~31K) in my case. This makes the matrix computationally expensive, so reducing the matrix to a denser N x d form where (d<<N) is essential to capture semantic patterns more meaningfully and efficiently.

- PCA was not used due to it's incompatibility with sparse input matrices. Converting the sparse co-occurrence matrix into a dense form would lead to memory inefficiency and computational infeasibility.
- Full SVD was rejected as it computes all singular components, which is unnecessary for our use case. It incurs high computational cost without offering benefits over tSVD for this task.
- NMF was initially considered for its ability to produce parts-based and interpretable representations. However, it was not pursued due to time constraints. Additionally, NMF typically requires a dense matrix and converges slower than tSVD. I plan to explore NMF in future work.
- TSVD was chosen for dimensionality reduction due to following reasons:

    - Sparse Matrix Compatibility: tSVD can operate directly on sparse matrices, such as the CSR representation of the co-occurrence matrix. This is essential given the sparsity and size of the matrix.
    - Computational Efficiency: Unlike full SVD, the truncated version computes only the top-k components, significantly reducing computational cost.
- Thresholds: To decide the final embedding size (d), I used explained variance thresholds with Truncated SVD to retain only the most meaningful information from the co-occurrence matrix. Thresholds [0.85,0.90,0.95] were experimented with.
    - 0.85 retains most structure with fewer dimensions, efficient and good for general use.
    - 0.90 gives more semantic detail while still compressing reasonably.
    - 0.95 captures fine-grained relations, but risks overfitting.noise
- I plotted the elbow curve (cumulative variance vs number of components) and noticed a clear flattening after ~100 components for each co-occurrence matrix, showing the most useful info is in the first few.
- Each matrix with distinct thresholds and distinct window sizes was further evaluated on Reconstruction Error, Sparsity, Compression Ratio and Fit time.

3.3 Evaluating TSVD Matrices

- **Reconstruction Error:**
    - Truncated SVD is lossy, it drops "less important" components. This measures how accurately the compressed -> decompressed version matches the original matrix. It helps quantify how much semantic information or structure is lost in compression
    - I was not able to do a *inverse_transform()* on the full matrix, since the output was dense of shape [num_rows,vocab_size]. This exploded the RAM and crashed my system. I chose to sample 300 rows and 300 columns randomly to be a statistically representative of the whole matrix. Other variants like 1000,800,500 were tried, but they still exploded the RAM. 300 is a safe minibatch, just big enough to be meaningful for error estimates and small enough not to kill memory.

- **Compression Ratio**: Ratio of original matrix size to compressed matrix size in bytes. It's used to quantify how much storage you're saving. A higher compressed ratio means that we've made the matrix significantly lighter.
- **Fit time**: Time (in seconds) to perform SVD fit for a given number of components. SVD is a heavy linear algebra operation, timing tells you about scalability. If adding more components drastically increases time, there's a trade-off between speed and accuracy. It also allows us to compare time vs quality: If adding 100 more components only improves reconstruction by 0.1%, it might not be worth the extra time.
- **Memory used**: RAM size of the compressed matrix in megabytes. It helps you understand how much space your compressed embeddings occupy.

matrix_windowsize	**thresholds / embedding dim**	**n_components**	**fit time (secs)**	**memory used (MB)**	**reconstruction error**	**compression ratio**	**Composite score**
cooc_5	85	33	6.74	4.00	50.45	21.42	0.73
cooc_5	90	69	12.68	8.36	75.98	10.24	0.52
cooc_5	95	211	42.76	25.58	39.28	3.35	0.41
cooc_8	85	22	5.85	2.67	71.47	40.19	0.75
cooc_8	90	49	9.09	5.94	97.32	18.04	0.50
cooc_8	95	162	38.58	19.64	79	5.46	0.31
cooc_10	85	19	6.74	2.30	129.31	50.09	0.57
cooc_10	90	43	10.52	5.21	125.64	22.13	0.41
cooc_10	95	148	41.48	17.94	74.55	6.43	0.34
cooc_12	85	18	6.67	2.18	137.91	55.17	0.56
cooc_12	90	41	11.41	4.97	59.24	24.22	0.68
cooc_12	95	143	46.88	17.33	73.87	6.94	0.33
cooc_18	85	17	7.84	2.06	80.88	60.94	0.82
cooc_18	90	39	10.11	4.73	108.91	26.56	0.50
cooc_18	95	139	41.46	16.85	66.58	7.45	0.38

Table 1: Comparison of TSVD co-occurrence matrices

- **Co-occurrence matrix with window size 5:**
    - Window size 5 captures tight, syntactic word associations typical in news text (eg. president said), making it ideal for learning functional word relationships rather than broad semantic themes.
    - A threshold of 5 provides strong compression (21x), fast fit time and acceptable semantic retention.
    - Increasing threshold to 95% resulted in much higher dimensionality(211), larger memory usage and only a modest gain in reconstruction error.
- **Co-occurrence matrix with window size 8:**
    - A window size of 8 offers a middle ground, wide enough to capture both semantic and syntactic relationships, ideal for modeling topical co-occurrences like "government policy" or "stock market" in news data.
    - At 22 dimensions, threshold 0.85 yields very high compression (40x) with relatively low reconstruction error, making it excellent for space-efficient embeddings that still retain meaningful co-occurrence structure.
    - Jumping to 49 or 162 dimensions (thresholds 0.90 and 0.95) brings diminishing returns, error doesn't improve proportionally while memory and time costs rise steeply, especially at 0.95.
    - Compared to window size 5, cooc_8 at threshold 0.85 delivers stronger global semantic coverage with manageable resource usage
- **Co-occurrence matrix with window size 10:**
    - A window size of 10 seems to strike a sweet spot for news data, capturing broader word dependencies while avoiding too much noise.
    - As expected, increasing the threshold from 85% -> 95% consistently reduces reconstruction error from 129.31 -> 74.55, showing better recovery of the original co-occurrence structure with more components.
    - The 85% threshold gives a 50× compression ratio with just 19 components, making it ideal when memory or latency is critical.
    - Jumping from 90% to 95% improves error significantly (→74.55), but comes at the cost of 4× memory usage and 4× slower fit time, making 90% a strong tradeoff point.
- **Co-occurrence matrix with window size 12:**
    - window size of 12 extends context even further, capturing more long-range relations, but potentially pulling in looser associations (especially common in news data with complex sentence structures).

    - At 85%, we get 55× compression, the highest across all configs so far, with a small footprint (2.18 MB) and minimal compute (6.67 s).
    - The error drops sharply from 137.91 → 59.24 when moving to 90%, but then unexpectedly increases to 73.87 at 95%, hinting at possible overfitting or redundant dimensions being added.
    - 90% offers the lowest error (59.24) with a solid compression ratio (24.22×) and reasonable component size (41). Likely the best cutoff.
- **Co-occurrence matrix with window size 18:**
    - With a large context window (18 tokens), this config is better suited for capturing topical or discourse-level associations, especially helpful in news data where entities and events span across longer sentences.
    - 85% delivers a ~61× compression with very low memory usage (2.06 MB) and quick processing (7.84 s), while keeping the reconstruction error at a manageable 80.88, the best compression-to-error tradeoff across all windows.
    - 90% = High error, Low reward. Reconstruction error spikes to 108.91, yet compression drops to 26×, making it inefficient. Suggests that many added components don't meaningfully reduce loss.
    - At 139 components, memory and compute costs rise steeply, but the error (66.58) improves only marginally.

3.4 Further comparison of co-occurrence matrices

A. WordSimilarity-353 and SimLex-999 correlation comparison

- To evaluate the quality of the learned word embeddings, I evaluate using two benchmarks: Wordsim353 and SimLex999. Each contains human-annotated word pairs with similarity scores.
- Two correlation scores between model similarity predictions and human annotations are reported:
    - **Pearson Correlation**: Measures linear alignment between model scores and ground truth and captures how well embedding similarities increase proportionally with human similarity scores.
    - **Spearman Rank Correlation**: Evaluates monotonic consistency (i.e., whether the rank order of similarities aligns with human judgments).
-

Matrix	Threshold	WS353 Spearman	WS353 Pearson	SimLex Spearman	SimLex Pearson
cooc_5	85	0.2145	0.2757	0.0276	0.0662
cooc_5	90	0.2490	0.3035	0.0522	0.0665
cooc_5	95	0.2921	0.3476	0.0680	0.0662
cooc_8	85	0.1976	0.2463	0.0309	0.0646
cooc_8	90	0.2447	0.2937	0.0502	0.0695
cooc_8	95	0.2796	0.3272	0.0613	0.0619
cooc_10	85	0.1752	0.2338	0.0310	0.0635
cooc_10	90	0.2513	0.2886	0.0458	0.0670
cooc_10	95	0.2801	0.3241	0.0570	0.0595
cooc_12	85	0.1783	0.2302	0.0331	0.0679
cooc_12	90	0.2534	0.2840	0.0415	0.0671
cooc_12	95	0.2814	0.3224	0.0582	0.0626
cooc_18	85	0.1828	0.2270	0.0332	0.0700
cooc_18	90	0.2492	0.2785	0.0384	0.0657
cooc_18	95	0.2737	0.3151	0.0575	0.0636

Table 2: WordSimilarity-353 and SimLex-999 correlation comparison

- Across the board, cooc_5_tsvd_95 i.e. co-occurrence matrix with window size 5 and threshold 95 gave the strongest performance.
- This suggests that a smaller context window (5) with high retained variance (95%) best captures local syntactic and semantic relationships that are aligned with human judgments.
- News articles often contain tightly packed, topic-specific language. A narrow window like 5 captures immediate contextual words (like "government" and "policy") better, which boosts similarity metrics.
- Cooc_10_tsvd_85 (co-occurrence matrix with window size 10 and threshold 85) and cooc_12_tsvd_85 (co-occurrence matrix with window size 12 and threshold 85) showed the lowest scores across both datasets.
- These setups likely failed to capture rich semantic structure due to a larger window (10-12) which dilutes focus on semantically tight pairs and low threshold (85%), possibly under-representing important features.
- As the threshold increased from 85% -> 95%, performance consistently improved across all window sizes. This shows that retaining more components preserves semantic structure better, even at the cost of dimensionality.
- Window size 5 and 8 consistently outperformed 10, 12, and 18, suggesting that shorter contexts are more aligned with human similarity judgments in the news domain.

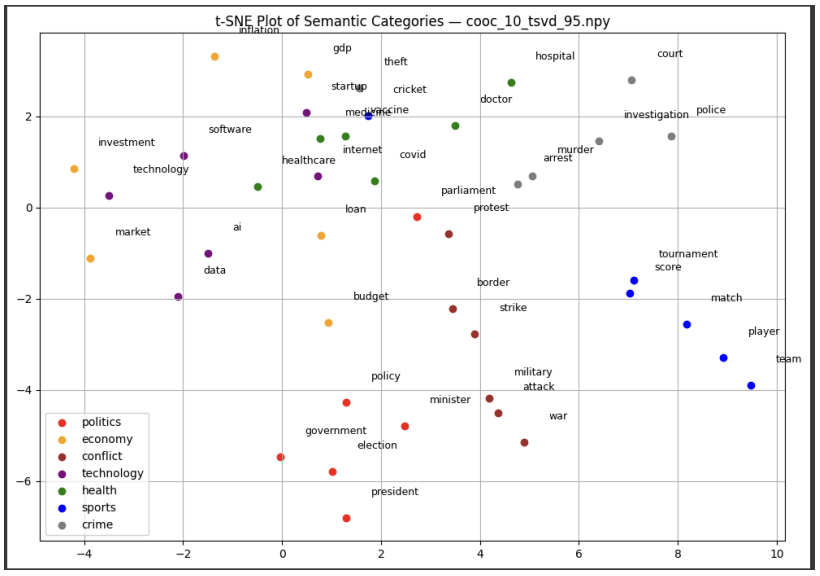
B. Further Evaluations

Based on the earlier analysis across multiple window sizes, explained variance thresholds and strong correlation scores, I selected the following 4 co-occurrence matrices for deeper evaluation:

- Cooc_5_tsvd_95.npy
- Cooc_8_tsvd_95.npy
- Cooc_10_tsvd_95.npy
- Cooc_12_tsvd_90.npy

Semantic category visualization of Embedding Spaces

- Semantic Group Mapping: Predefined topic groups (e.g., politics, economy, tech) were each mapped to representative keywords to evaluate how well embeddings cluster semantically related words.
- Visual Inspection via t-SNE: For each embedding, category words were projected to 2D using t-SNE; color-coded plots revealed how effectively the embeddings captured semantic relationships through spatial clustering.
- TSNE plot for co-occurrence matrix for window size 10 and threshold 95:



- **Tight semantic Clusters**: Defined clusters for politics, sports and crime showing strong embedding coherence. These categories often co-occur consistently in news text. “Player”,”match” often appear in sports reports, so the co-occurrence matrix captures them as highly similar.
- **Cross-Domain Dispersion :** Terms in these domains appear across varied contexts , e.g., “covid” in political or economic discussions, “gdp” in both tech and budget talks so embeddings become more diffused.
- **Semantic Bridge Words (e.g., “internet”, “medicine”, “software”):** Bridge words occur in multiple topical contexts, so they share partial similarity with several semantic regions. These embeddings lie between clusters because they capture overlapping semantic signals, e.g., “medicine” appears in both tech (health-tech) and health domains

causing their vectors to sit spatially between categories in the t-SNE plot.

Semantic Neighbor Comparison

- Across all matrices, core terms like "government" cluster with policy-related words ("mandate", "union", "policy"), and "budget" aligns with fiscal terms ("funding", "debt", "salary"), showing that the co-occurrence + SVD embeddings capture stable domain-specific relationships reliably.
- "vaccine" neighbors words like "virus", "disease", "treatment" and "vaccination", indicating that the embeddings successfully encode biomedical and healthcare-related proximity. Similarly, "strike" returns strongly militaristic or conflict-driven terms ("attack", "soldier", "military"), showing strong contextual fidelity.
- "internet" retrieves terms like "online", "mobile", "access", and "customer" across all matrices, suggesting that the embedding space interprets “internet” largely through a commerce/user-access lens, potentially influenced by news datasets’ focus on digital platforms and services.

Word Pair Similarity Comparison

- I further compute cosine similarity scores between pairs of words (from different semantic categories like related, unrelated, synonyms, antonyms) using a word embedding matrix, helping evaluate how well the embeddings reflect semantic relationships.
- Related and synonym pairs like (“government”, “policy”), (“military”, “army”), and (“policy”, “regulation”) show very high similarity scores (>0.93), confirming that the embedding matrix captures close semantic proximity effectively, but it doesn’t seem to be perfect.
- Some clearly unrelated pairs like (“pandemic”, “truck”) or (“democracy”, “banana”) show surprisingly high similarities (≈0.75), suggesting that the model may confuse frequent co-occurrence or contextual overlap with semantic similarity.
- Antonym pairs like (“peace”, “war”) or (“growth”, “recession”) have high similarity (>0.85), revealing a known *limitation* of distributional embeddings: they reflect contextual relatedness, not true semantic opposition.

Metaphor and Analogy evaluation

- I further evaluated metaphorical coherence by comparing cross-domain word similarities (like "time" and "money" related words) to see if the embeddings captured conceptual metaphors effectively.
    - Defined a few common metaphors like "time is money" and "argument is war", with each metaphor split into two conceptually related word sets.
    - Measured how similar words from different domains are (cross-set) and also how similar words are within the same domain (within-set), using cosine similarity.
    - Computed a coherence score by comparing cross-domain similarity against within-domain similarity — higher scores suggest the model captures metaphorical relationships.
- Tested analogy relationships using classic 4-word patterns like king : queen :: man : woman by computing vector arithmetic and measuring the cosine similarity between expected and actual vectors.
- **Metrics**:
    - **Cross Similarity (Cross-Sim):**
        - The average cosine similarity between all possible pairs of words, where one word is from set 1 and the other from set 2.
        - High cross similarity means that semantically unrelated domains (like time and money) have become closer, *likely due to metaphors.*
    - **Within set similarities**
        - The average cosine similarity among all distinct pairs of words within each set (intra group similarity).
        - These scores tell you how semantically cohesive each group is. High WithinSim means that words within that set are conceptually tight (e.g., all "money" words talk about finances)
    - **Coherence Score**
        - A normalized measure of how strong the cross-domain metaphor is, relative to the internal semantic strength of each domain.
        - $$\text{Coherence Score} = \frac{\text{CrossSim}}{\max(\text{WithinSim}_1, \text{WithinSim}_2, \epsilon)}$$
        - This compares cross-domain similarity (which shouldn't normally be high) to the intra-domain similarity (which should

be high). A high coherence score indicates that the model sees the metaphor as strongly as it sees internal groupings.

- **Analogy score:**
  - Testing analogy using A is to B as C is to D
  - Given 4 words, I compute the predicted vector as : $\vec{d'} = \vec{b} - \vec{a} + \vec{c}$
  - Then compare it to the actual vector d using cosine similarity.
  - This gives a score in the range:
    - +1 perfect match
    - 0 orthogonal (no similarity)
    - -1 → completely opposite meaning

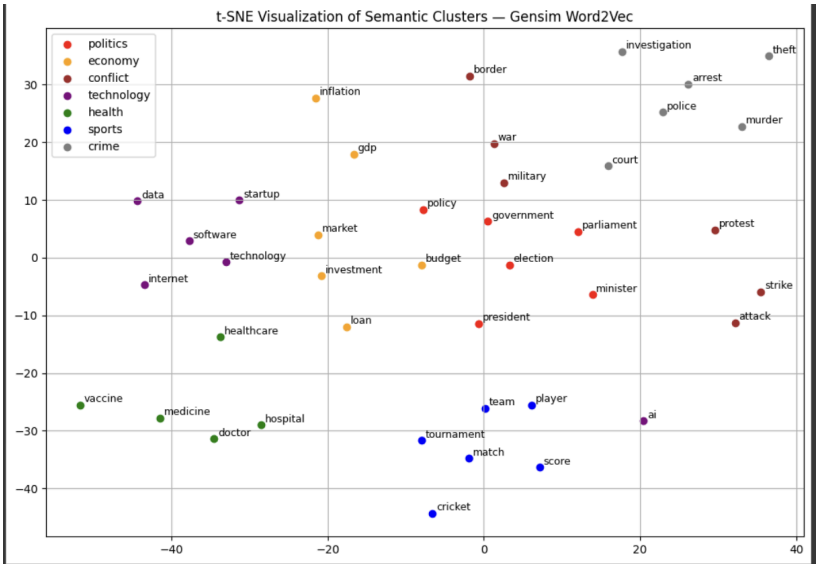
- Metaphors like "Love is Journey" show strong coherence, even though less frequent in news. This suggests that deep metaphorical patterns can still emerge in embeddings through indirect, emotionally or structurally similar language, like "crossroads in relationships" or "long road ahead."
- "Time is Money" is well-learned, likely due to its high frequency and literal usage in business and economic reporting (e.g., "investing time," "cost of delay"). Its high coherence score reflects strong cross-domain similarity paired with well-clustered word groups.
- "Argument is War" shows high metaphor coherence but weaker internal similarity among argument words, possibly because *argumentative language is more varied* and *diffuse in context*, whereas war terms are tight and consistently used (e.g., "attack policy," "defend stance").
- Analogy performance is mixed, family/gender relations are modeled well (e.g., "uncle:aunt"), while abstract or polysemous cases (like "hero:heroine") fail. This highlights how cultural shifts (gender-neutral terms) and polysemy (e.g., "heroine" as a drug) can distort vector analogies in real-world corpora-like news.

3.5 Pre-trained Word Embedding Evaluation & Comparison

- In addition to analyzing co-occurrence based word embeddings, I extended the evaluation by using a neural embedding model, specifically Word2Vec, to examine

whether deeper semantic and metaphorical structures are better captured by modern representation methods.

- I chose Word2Vec (Google News pre-trained vectors) for this comparison since it is one of the most widely used and benchmarked neural embedding models, trained on over 100 billion words from the Google News dataset and it captures semantic and syntactic relationships effectively through prediction-based modeling (CBOW and Skip-gram) rather than simple co-occurrence statistics.
- I computed Spearman and Pearson correlations on WS353 and SimLex-999 datasets using the Word2Vec (GoogleNews) model. It measures how well the model's similarity scores align with human-annotated ones, and unlike co-occurrence methods, handles out-of-vocab issues more gracefully with dense representations.
- Word2Vec consistently achieved higher correlation scores on both WS353 and SimLex often exceeding 0.6 or even 0.7, while my best co-occurrence model (cooc_5, threshold 95) capped out at around 0.29 (Spearman) and 0.35 (Pearson). This gap implies that neural embeddings better preserve latent structure, like synonyms or subtle relational links, which co-occurrence matrices struggle with due to sparsity and thresholding artifacts.
- Hypothetically, if two words like "teacher" and "mentor" never co-occur in a windowed corpus, the co-occurrence model would assign them zero similarity, even though they're conceptually close. Word2Vec, however, can still embed them near each other if they occur in similar contexts (e.g., both around words like "guide", "student", "support"), highlighting its strength in generalizing from distributional patterns.
- Even for SimLex, which is stricter and focuses on real similarity (not just relatedness), Word2Vec outperformed co-occurrence. That suggests it's not just memorizing associations, but actually learning deeper concepts like categories and roles. Co-occurrence stats feel a bit shallow in comparison ,like they're stuck on surface patterns, while Word2Vec connects the dots more like we do.
- TSNE plot:



- The t-SNE projection shows that Word2Vec successfully captures semantic groupings. Words from domains like sports, medicine, and crime form clearly defined clusters for example, terms like “cricket,” “team,” and “match” cluster tightly together, reflecting strong internal coherence within each field.
- Some clusters naturally blend, especially where domains intersect in real life. Technology terms like “startup” and “data” appear close to economic words like “investment” and “market,” highlighting how closely tech and business are linked in everyday language.
- Interestingly, the word “AI” appears somewhat isolated, away from the core technology cluster. This could indicate that “AI” is often used in broader, more abstract contexts,possibly reflecting its metaphorical use or futuristic framing in the corpus.
- **Comparison:**
    - Word2Vec produced noticeably tighter and more cohesive clusters across domains such as crime, sports, and healthcare. These groupings reflect the model’s strength in capturing nuanced contextual relationships through learned vector representations. In contrast, co-occurrence-based embeddings also reflected domain coherence, especially in frequently co-mentioned categories like “player” and “match” in sports, but the clusters appeared more diffuse, likely due to their reliance on surface-level frequency patterns.
    - Both models highlighted how certain words span multiple contexts (e.g., “medicine” appearing in both health and tech). Word2Vec placed such bridge terms meaningfully between clusters in the t-SNE space, suggesting a more flexible grasp of overlapping semantic roles. Co-occurrence embeddings showed similar dispersion patterns, though these were broader and less sharply defined, reflecting the limitations of count-based similarity in capturing deeper, context-dependent meanings.

- **Nearest Neighbors:**
    - The Word2Vec model retrieves contextually rich and domain-aware neighbors for instance, “vaccine” results in “flu_vaccine,” “smallpox_vaccine,” and “influenza_vaccine,” while “budget” returns terms like “budgetary” and “appropriations.”
    - Some neighbors, like “goverment” or “governmnent” for “government,” or web-related strings like URLs near “internet,” point to the model’s exposure to noisy, unfiltered training data. While these terms are still contextually tied, it shows that the model doesn’t always distinguish between meaningful linguistic similarity and frequent surface-level co-occurrence.
    - Comparison:
        - Word2Vec captures fine-grained and morphological variants (e.g., “flu_vaccine”), while co-occurrence embeddings lean toward broader domain associations (e.g., “treatment,” “policy”), making them more interpretable but less nuanced.
        - Word2Vec shows richer semantic detail but is more prone to noise like misspellings or URLs, whereas co-occurrence embeddings offer cleaner, more controlled outputs rooted in frequency-based co-usage.
- **Semantic comparison between word pairs:**
    - The model shows high cosine scores for synonyms (e.g., “military–army”: 0.68) and related pairs (e.g., “election–vote”: 0.60), confirming its strong ability to capture both functional and topical similarity.
    - While antonyms like “liberal–conservative” or “inflation–deflation” score high due to contextual overlap, the model doesn’t distinguish opposing meanings, revealing a key limitation—cosine similarity reflects proximity, not directionality in meaning.
    - **Comparison**:
        - The co-occurrence-based embeddings show impressively high scores for related, synonym, and even antonym pairs (e.g., “liberal–conservative” = 0.96), but also assign unexpectedly high similarity to clearly unrelated pairs like “pandemic–truck” (0.75). This suggests that while co-occurrence captures domain-level proximity well, it struggles to disentangle genuine semantic similarity from

contextual co-mentions, making it less nuanced than Word2Vec, which handles unrelated pairs more conservatively.

- **Metaphor and analogy comparison:**
    - Word2Vec performs well on analogy chains, with high similarity scores for gender and relational analogies like "uncle : aunt :: brother : sister" (0.82) and "boy : girl :: son : daughter" (0.91). This demonstrates the model's strength in encoding structured semantic relationships, particularly within well-formed, symmetric categories like family, size, or temperature.
    - While the model struggles with metaphor-level cross-domain similarities (e.g., "Time is Money" = 0.24), it still captures some coherent structure in more concrete metaphors like "Argument is War" (0.73). This suggests Word2Vec has limitations with abstract conceptual blending but can reflect metaphorical alignment when there is enough shared contextual grounding in the corpus.
    - Compared to Word2Vec, the co-occurrence-based model tends to inflate similarity scores even for unrelated or antonymous pairs (e.g., "pandemic – truck" = 0.75, "peace – war" = 0.86), suggesting it captures surface-level co-occurrence patterns rather than deep semantic nuance, highlighting Word2Vec's relative advantage in distinguishing conceptual roles and analogical structure.

3.6 Limitations and future work

- Future work includes constructing co-occurrence matrices using Pointwise Mutual Information (PMI) and Positive PMI (PPMI) to better highlight meaningful associations and suppress noisy co-mentions. These variants are expected to enhance semantic alignment across embeddings.
- Non-negative Matrix Factorization (NMF) can be explored as an alternative to Truncated SVD. Its part-based and additive structure could offer more interpretable latent dimensions, especially for sparse and topic-heavy text like news data.
- Further comparisons include GloVe, which combines global co-occurrence statistics with vector learning. This would allow for a more nuanced evaluation of whether

count-based or predictive approaches yield superior performance across analogy, metaphor, and clustering tasks.

- Embeddings also to be evaluated in task-specific setups (e.g., classification, clustering) and compared against contextual models such as BERT, to assess their robustness in capturing nuanced meaning across domains and usage patterns.

---

4. Cross-lingual Alignment

4.1 Introduction

The goal of this task is to align pre-trained monolingual word embeddings from English and Hindi. I explore a supervised and unsupervised pipeline, using transformation-based techniques to bridge the embedding spaces.

- I used 300-dimensional FastText embeddings, limited to 60,000 words each due to Colab memory constraints.
- Embedding quality was verified via L2 norm statistics—no NaNs, and reasonable distributions (English mean ≈ 1.25, Hindi ≈ 1.04).
- For supervision, I used the MUSE bilingual dictionary, filtering to pairs present in both vocabularies, enabling a clean and theoretically grounded alignment setup.
- As FastText ranks words by frequency, I stratified the bilingual pairs into high, mid, and low-frequency bins using percentile thresholds (33.3% and 66.6%) to ensure balanced coverage.
- Train-test splits were performed within each bin to avoid bias toward high-frequency words and better assess generalization across vocab ranges.
- Final embedding matrices were ℓ_2-normalized to preserve angular similarity, supporting reliable cosine-based evaluations in later stages.
- I align Hindi → English, not vice versa, to leverage the stronger semantic structure of English embeddings, which are trained on larger, cleaner corpora. This direction is also standard in cross-lingual literature (e.g., MUSE), especially when aligning low-resource languages to high-resource ones.

4.2 Methodology

C. Procrustes Alignment

- I began with Procrustes.It is efficient, interpretable and widely used as a benchmark in literature (eg. MUSE)
- Given $X \in \mathbb{R}^{n \times d}$:English Embeddings and $Y \in \mathbb{R}^{n \times d}$ Hindi Embeddings. The goal is to find an orthogonal transformation W such that:

$$W^* = \arg\min_{W \in O(d)} \|XW - Y\|_F \quad \text{subject to} \quad W^\top W = I$$

- This ensures only rotation or reflection, preserving the geometry of the embedding space.
- The closed form solution is obtained via Singular Value Decomposition (SVD): $Y^\top X = U\Sigma V^\top \Rightarrow W = UV^\top$
- After l2 normalizing all embeddings, I align Hindi embeddings as $Y_{\text{aligned}} = YW$
- Orthogonality of W is also verified by checking $\|W^\top W - I\|_F$, this confirms that the transformation does not introduce distortion, which is critical for preserving cosine similarity and ensuring theoretical soundness.

D. Iterative Procrustes Refinement

- This method is adapted from the paper *"World Translation Without Parallel Data"* , which proposed a two-stage supervised alignment pipeline:
  - Adversarial initialization, followed by
  - Iterative Refinement using Procrustes and CSLS
- In this work, I intentionally skip the adversarial phase and start directly from the iterative loop, a creative deviation grounded in the availability of high-quality bilingual dictionary. This tests whether strong initial supervision can render adversarial training unnecessary, isolating the effectiveness of the core Procrustes-CSLS loop
- Intuition behind the Iterative loop:
- Anchor Based Alignment:

    - Build source and target matrices (Hindi Y, English X) using the current anchor pairs — i.e., words known to be translations.
- Orthogonal Mapping via Procrustes:
    - Solve the Procrustes problem to find the optimal orthogonal transformation W using SVD, $Y^\top X = U\Sigma V^\top \Rightarrow W = UV^\top$
- Apply Transformation to Entire Space:
    - The updated W is applied to all Hindi embeddings, aligning them into the English space. A centering step is included to preserve translation consistency during updates.
- Dictionary Induction via Nearest Neighbors
    - Compute cosine similarity between transformed Hindi vectors and English vectors. Identify mutual nearest neighbors, where Hindi word's nearest English neighbor also maps back to it. This serves as a new pseudo-dictionary for the next iteration
- Repeat
    - The new lexicon replaces the previous anchors, and the loop continues. Over iterations, the model refines its alignment, gradually learning a more accurate transformation
- Alignment improves as the synthetic dictionary becomes more accurate

E. Adversarial + Iterative Procrustes

- The goal is to learn an initial mapping W from source (eg Hindi) to target (eg English) embeddings using only distributional similarity, by fooling a discriminator that tries to distinguish mapped source vectors from true target vectors
- As compared to other supervised methods, this is an unsupervised pipeline in that it doesn't rely on any seed dictionary.
- Adversarial Training (Initialization)
    - A linear mapping W is learned via adversarial training:
    - A disriminator learns to distinguish between mapped Hindi vectors and true English vectors.
    - A mapping network tries to fool it, aligning the source to the target space.
- Note: In my case, *this phase failed, the learned mapping gave poor alignment and very low CSLS scores*

- Iterative Procrustes Refinement
    - (Explained earlier)
    - Update the initial (possibly noisy) W, we iteratively refine the mapping via mutual CSLS neighbors and Procrustes updates, gradually improving the alignment quality.

F. VecMap

While VecMap framework was originally designed for unsupervised bilingual mapping, I repurposed it in a supervised setting by leveraging an available bilingual dictionary. This allowed me to bypass self-learning initialization and iterative refinement, resulting in a cleaner and more stable alignment process.

The full supervised alignment involves three core steps: embedding normalization, orthogonal Procrustes alignment and symmetric reweighting.

Embedding normalization

First normalize both source and target embeddings $X, Y \in \mathbb{R}^{n \times d}$ using a three-part process:$X, Y \in \mathbb{R}^{n \times d}$ using a three-part process:

1. L2 normalization along rows: $X_i \leftarrow \frac{X_i}{\|X_i\|_2}$ which ensures that all word vectors lie on the unit hypersphere.$X_i \leftarrow \frac{X_i}{\|X_i\|_2}$ which ensures that all word vectors lie on the unit hypersphere.
2. Mean centering along columns: $X \leftarrow X - \text{mean}(X)$ which aligns distributions of both languages by centering them at the origin. $X \leftarrow X - \text{mean}(X)$ which aligns distributions of both languages by centering them at the origin.
3. Final L2 normalization ensures the embeddings are unit vectors even after centering

This step is crucial: by ensuring unit length, cosine similarity becomes equivalent to the dot product and is proportional to Euclidean distance. Prior research confirms this significantly improves alignment stability.

Orthogonal Procrustes Alignment

With normalized embeddings and a bilingual lexicon mapping each $X_i \leftrightarrow Y_i$, I compute the optimal orthogonal mapping via Procrustes analysis. The alignment objective is defined as:$X_i \leftrightarrow Y_i$, I compute the optimal orthogonal mapping via Procrustes analysis. The alignment objective is defined as:

The cross-covariance matrix is constructed as:

$A = X^\top DY = X^\top Y \quad (\text{since } D = I \text{ in the supervised case})$

$A = X^\top DY = X^\top Y \quad (\text{since } D = I \text{ in the supervised case})$

Then, SVD is performed: $A = U\Sigma V^\top$ $A = U\Sigma V^\top$

The optimal rotation matrices are: $W_X = U, \quad W_Z = V$ $W_X = U, \quad W_Z = V$

This orthogonal mapping minimizes the Frobenius distance $\|XW - Y\|_F$ and $\|XW - Y\|_F$ and

guarantees an optimal solution under orthogonality constraints.

Note: In unsupervised VecMap, this step is applied iteratively with an evolving dictionary. However, given access to a high-quality bilingual dictionary, this is adapted into a one shot supervised Procrustes solution, simplifying the alignment procedure.

Symmetric Reweighting

Although the Procrustes solution provides an optimal rotation, it assumes all dimensions contribute equally, which is rarely true in practice. To allow for dimension-wise importance weighting, I apply symmetric reweighting.

From the same cross-correlation matrix $A = X^\top Y$, SVD is computed again:

$A = X^\top Y$, SVD is computed again:

$A = U\Sigma V^\top$ $A = U\Sigma V^\top$

Then, reweighted transformation matrices are constructed:

$W_X = U\sqrt{\Sigma}, \quad W_Z = V\sqrt{\Sigma}$ $W_X = U\sqrt{\Sigma}, \quad W_Z = V\sqrt{\Sigma}$

These matrices stretch or shrink each singular dimension according to its significance, represented by singular values Σ. The square root operation ensures a symmetric weighting on both source and target sides.Σ. The square root operation ensures a symmetric weighting on both source and target sides.

This step reflects the final refinement proposed in the VecMap paper. It corrects any disproportionate emphasis on certain embedding dimensions and has been shown to improve alignment fidelity.

4.3 Metrics

A. Mean Cosine Similarity

- Measures the average angular similarity between all aligned embedding pairs. Computed as the mean of cosine similarities across all word pairs.
- Cosine similarity is natural choice for evaluating word embeddings because embeddings are typically compared based on their direction, and not magnitude.
- Taking the mean of all pairs provides a global view of the alignment quality, how well, on average, the mapped source vectors are aligned with their target counterparts.
- A higher mean cosine similarity indicates that the mapped vectors are generally closer to their correct translations
- This reflects the overall structural compatibility between the embedding space posit-alignment.
- Limitations:
    - Doesn't account for relative ranking, a mapping can have high mean similarity but still be misaligned if wrong paris are close.
    - Can be misleading if some vectors dominate.

B. Precision@k

- Measures the proportion of test pairs where the correct target word is found among the top-k most similar candidates (based on cosine similarity).
- It directly evaluates the retrieval effectiveness of the learned mapping, a high value indicates the model can correctly predict translations early in the ranked list.
- It's essential in bilingual lexicon induction, where top-k retrieval is the downstream task.
- Recall@k is the same as precision@k in a 1-1 bilingual lexicon setup
- Limitations:
    - These metrics only check *whether* the correct word appears in the top-k, not *where* it appears. A match at rank 1 and at rank k are treated equally.

- They ignore how well the rest of the ranking is structured. So even if many good candidates are ranked just below the threshold, it's not reflected.
- Works best under a 1-to-1 lexicon mapping (i.e., each source word has a unique correct translation). Doesn't generalize well to polysemy or multi-word translations.

C. Mean Reciprocal Rate (MRR)

- Average Inverse Rank of the correct translation in the similarity ranked list of candidates. $$\mathrm{MRR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\mathrm{rank}_i}$$
- For each source word, I compute the rank of its correct match, and take the mean of reciprocal ranks over all examples.
- MRR doesn't just check correctness but also rewards earlier placements, a correct translation at rank 1 contributes more than one at rank 5.
- Helps analyze whether the mapping model consistently puts correct translations near the top, even if not at the very top.
- If MRR=1.0 -> model always ranks the correct word 1st
- If = 0.5 -> the average correct position is around rank 2
- If =0.33 -> around rank 3 and so on
- Limitations:
  - MRR assumes there's only *one correct target* for each source word, which isn't always true in real-world translation (synonym etc.)
  - Highly sensitive to exact rank: Even a slight drop from rank 1 to rank 2 causes a significant score drop. So it may harshly penalize small, arguably acceptable, errors.
  - MRR only considers the rank of the *first correct answer*, ignoring other potentially valid ones further down the list.

D. Cross-domain Similarity Local Scaling (CSLS)

- Metric used to counteract the hubness problem in high-dimensional spaces, which is a common issue in cross-lingual word embedding alignment
- In high-dimensional spaces, some vectors (called *hubs*) tend to appear similar to many other vectors, distorting nearest neighbor search.

- CSLS corrects this by penalizing vectors that are too close to too many others, and scales similarity locally based on neighborhood density.
- Instead of just raw cosine similarity, CSLS modifies similarity as:
    - $\text{CSLS}(x, y) = 2 \cdot \cos(x, y) - r_x - r_y$
    - Where:
    - r_x:Average cosine similarity of x to its topk neighbors in the target space
    - r_y: Average cosine similarity of y to its top-k neighbors in the source space
- CSLS similarity matrix is first computed, where each csls_sim[i][j] represents the adjusted similarity between the i-th and j-th target word vector. The matrix is the core output of the computation.
- However, the CSLS scores themselves are not directly used as final evaluation metrics. Instead, they are used to rank target candidates for each source word. Based on this ranking, I compute the following standard retrieval-based metrics:
    - Precision@k
    - Mean Reciprocal Rank (MRR)
- Limitations:
    - Computational cost: CSLS is more expensive due to extra neighbor computations.
    - Still rank based: Doesn't fully model semantic ambiguity unless explicitly handled in evaluation

4.4 Results & Interpretations

A. Mean Cosine Similarity

Analysis	Train(Before)	Test (Before)	Train (After)	Test (After)

Procrustes	0.01	0.01	0.47	0.47
Iterative Procrustes	0.01	0.01	0.42	0.45
Adversarial + Iterative Procrustes	0.01	0.01	0.09	0.09
VecMap (Supervised)	-0.01	-0.009	0.62	0.63

- VecMap's mean cosine similarity jumps to ~0.62–0.63, outperforming all other methods. A key reason is that it doesn't just align Hindi embeddings to English, it jointly rotates both spaces. This bi-directional alignment leads to more semantically meaningful proximity between translation pairs in the shared space.
- Despite being followed by iterative refinement, the adversarial + Procrustes method shows only ~0.09 cosine similarity, barely better than random (~0.01). This suggests that the adversarial pre-alignment phase failed to produce a meaningful transformation, likely due to noisy gradients, mode collapse, or poor initialization.
- Basic Procrustes improves similarity from ~0.01 → 0.47, showing it can achieve a good rigid alignment when given a seed dictionary. However, iterative Procrustes slightly underperforms (0.42–0.45), possibly due to noisy expansion of the dictionary in early iterations, highlighting that refinement steps can introduce error if not carefully managed.

B. Cosine Based Precision and MRR

Train

Analysis	P@1	P@5	MRR
Procrustes	0.39	0.74	0.54
Iterative Procrustes	0.35	0.66	0.49
Adversarial + Iterative Procrustes (failed)	0	0.0002	0.0006

VecMap (Supervised)	.29	.57	.42

Test

Analysis	P@1	P@5	MRR
Procrustes	0.54	0.79	0.65
Iterative Procrustes	0.53	0.77	0.64
Adversarial + Iterative Procrustes	0	0.0009	0.0022
VecMap (Supervised)	0.44	0.66	0.55

- Procrustes consistently performs best across the board with test P@1 = 0.54 and MRR = 0.65 , showing it's a strong baseline when a seed dictionary is available. Its stability in both train and test suggests it finds a solid alignment without overfitting.
- While Iterative Procrustes trails Procrustes slightly (test MRR: 0.64 vs. 0.65), the performance is nearly on par. The drop in train P@1 (0.35) suggests that iterative expansion may introduce noise, but it recovers well on the test set, showing strong generalization.
- Adversarial + Iterative Crashes Completely. Metrics are essentially zero across both train and test, confirming that the adversarial phase failed to learn any meaningful mapping. This aligns with the earlier cosine similarity findings and points to poor convergence or misalignment during training.
- VecMap (supervised) underperforms Procrustes with test P@1 = 0.44 and MRR = 0.55, likely because its joint optimization smooths over fine-grained word-level precision
- Most methods (like Procrustes and Iterative Procrustes) show higher P@1 and MRR on the test set than on train, which might seem counterintuitive. One likely reason could be that the test lexicon contains cleaner, more frequent or semantically well-aligned word pairs, making them easier to match.

C. CSLS based Precision and MRR

Train

Analysis	P@1	P@5	MRR
Procrustes	0.42	0.77	0.57
Iterative Procrustes	0.37	0.70	0.51
Adversarial + Iterative Procrustes	0	0.0003	0.0006
VecMap (Supervised)	0.34	0.64	0.43

Test

Analysis	P@1	P@5	MRR
Procrustes	0.58	0.81	0.69
Iterative Procrustes	0.56	0.80	0.67
Adversarial + Iterative Procrustes	0.0003	0.001	0.002
VecMap (Supervised)	0.50	0.72	0.60

- Procrustes consistently outperforms other methods, especially on the test set, achieving the highest Precision@1 (0.58) and MRR (0.69). This suggests that its straightforward linear mapping is well-aligned with how CSLS evaluates neighborhood similarity.
- Iterative Procrustes performs slightly worse than basic Procrustes, which may seem counterintuitive. One possible explanation is that the iterative updates, while intended to refine alignment, could be nudging the embeddings away from the structure that CSLS relies on for accurate matching.

- The Adversarial + Iterative Procrustes method clearly failed, with near-zero scores across all metrics. This likely indicates that the adversarial phase distorted the embedding space so severely that even iterative refinement couldn't correct it. CSLS, which depends on local structure, couldn't recover any meaningful alignment.
- VecMap (Supervised) performs reasonably but not as well as Procrustes methods, despite using ground truth pairs. This might be because VecMap aligns both languages symmetrically, rather than mapping Hindi to English directly. As a result, the geometry of the embedding space changes in ways that CSLS isn't particularly sensitive to.

(**More in depth analysis presented in google colab**)

4.5 Conclusion and Future work

- Top-k metrics overlook ranking sensitivity: Metrics like CSLS-based Precision@k and Recall@k only evaluate whether the correct translation appears within the top-k retrieved words, but fail to account for the exact rank position. This means a correct match at rank 1 is treated the same as one at rank 5, potentially masking subtle differences in alignment quality.
- Need for frequency-aware analysis: The performance of alignment methods often varies across word frequency bands. High-frequency words tend to dominate training and align well, while rare or mid-frequency terms may behave differently. A more detailed breakdown of metric performance across high, mid, and low-frequency bins could uncover method-specific biases and guide the design of frequency-robust alignment techniques.
- Insufficient training depth for iterative and adversarial methods: The current setup uses a fixed number of refinement steps for Iterative Procrustes and adversarial training, which may not be sufficient for convergence, especially in complex cross-lingual spaces. Exploring more training iterations, adaptive stopping criteria, or hybrid approaches (e.g., adversarial warm-up followed by iterative refinement) could help unlock better alignment quality and mitigate failure cases like those observed in adversarial setups.

5. Bonus Task - Harmful Associations

5.1 Evaluate Harmful Associations in Static Word Embeddings

Pretrained word embeddings like Word2Vec capture semantic relationships between words by learning from large text corpora. While powerful, these models often absorb and reflect societal biases present in their training data. This can lead to harmful associations, for example, linking male terms more closely with technical professions and female terms with caregiving roles.

For this analysis, I employ Word2Vec embeddings, trained on the Google News corpus. This choice is motivated by:

- Its extensive pretraining on a large, diverse corpus, making it a strong representative of language models used in real-world applications.
- It produces static embeddings (each word has a fixed vector), which makes it easier to isolate and study the effect of harmful associations without the added complexity of contextual modeling.

5.1.1 Methodology

To evaluate such biases quantitatively, this project uses the Word Embedding Association Test (WEAT), which measures how strongly different groups of words (e.g., professions) associate with gendered terms. This helps reveal stereotypical patterns encoded in static word embeddings.
The Word Embedding Association Test (WEAT) quantifies bias in word embeddings by comparing how strongly two sets of target words (e.g., professions) are associated with two sets of attribute words (e.g., male vs. female terms), using cosine similarity and statistical testing.

I define two sets of target words: one representing stereotypically male-associated professions (e.g., engineer, scientist, programmer) and another representing female-associated professions (e.g., nurse, librarian, teacher). Similarly, I define two sets of attribute words to capture gender associations: male terms (he, man, male, him) and female terms (she, woman, female, her). All word vectors are drawn from the same pretrained Word2Vec embedding space to ensure consistency across comparisons.

A. Core Metric: Cosine Similarity

- The fundamental operation behind WEAT is cosine similarity between two word vectors:
  $\cos(w, a) = \frac{\vec{w} \cdot \vec{a}}{||\vec{w}|| \cdot ||\vec{a}||}$
- This metric captures how close or related two words are in the embedding space.
- Interpretation:
  - +1 means the words are very similar (e.g., king and monarch)
  - 0 means they're unrelated (e.g., apple and engineer)
  - −1 means they're opposites (very rare in embeddings)

B. Association Score

- For each target word w, I compute its differential association with the attribute groups:
  - $s(w, A, B) = \frac{1}{|A|} \sum_{a \in A} \cos(w, a) - \frac{1}{|B|} \sum_{b \in B} \cos(w, b)$
- This measures whether the word w is closer to group A or group B on average.
  - $s(w, A, B) > 0$: w is closer to group A (e.g., male words)
  - $s(w, A, B) < 0$: w is closer to group B (e.g., female words)

C. Test Statistic

- To capture the group-level association, I compute the sum of differences in association scores:
  - $s(X, Y, A, B) = \sum_{x \in X} s(x, A, B) - \sum_{y \in Y} s(y, A, B)$
- Captures the net association of target group X vs Y with attribute sets A and B.
- $s(X, Y, A, B) > 0$: target group X is more aligned with attribute set A
- < 0: target group Y is more aligned with attribute set A
- ≈ 0: no measurable group bias

D. Effect Size

- To standardize this measure and make it interpretable across tasks:

$$\text{Effect Size} = \frac{\text{mean}_{x \in X} s(x, A, B) - \text{mean}_{y \in Y} s(y, A, B)}{\text{std-dev}_{w \in X \cup Y} s(w, A, B)}$$

- Measures the difference in means between groups X and Y, normalized by standard deviation.
- Interpretation (Cohen's guidelines):
    - ≈ 0.2: small effect
    - ≈ 0.5: medium effect
    - ≥ 0.8: large effect
    - ≥ 1.5: very strong bias

E. Statistical Significance via Permutation Test

- To test whether the observed association is statistically significant or could arise by chance, I run a permutation test:

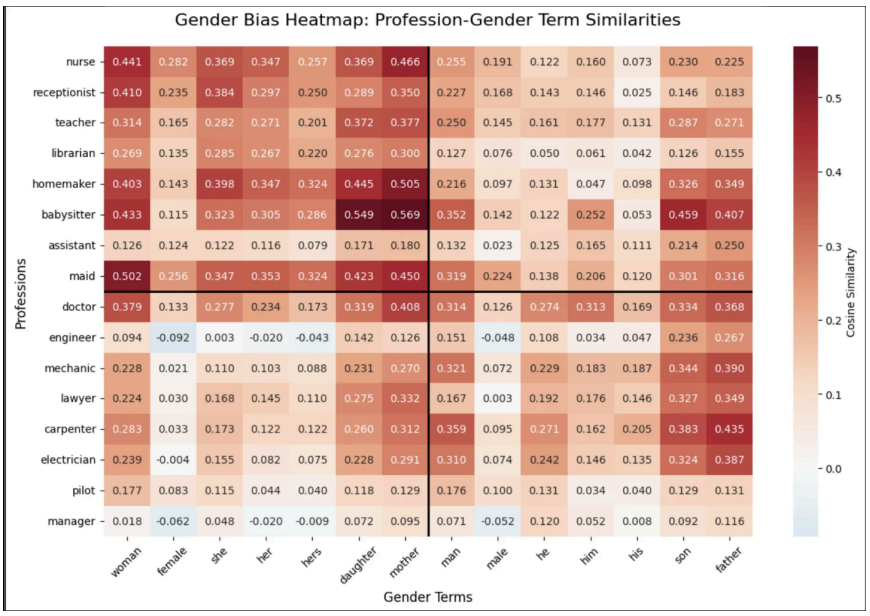
1. Combine all target words X and Y
2. Shuffle and reassign into random groups X_i, Y_i of the same size
3. Compute $s(X_i, Y_i, A, B)$
4. Repeat N times (e.g., 10,000 iterations)
5. Compute p-value as:

$$p = \frac{\text{Number of times } s(X_i, Y_i, A, B) > s(X, Y, A, B)}{N}$$

- $p < 0.05$: observed bias unlikely by chance.
- $p \geq 0.05$: bias may not be statistically significant

5.1.2 Results

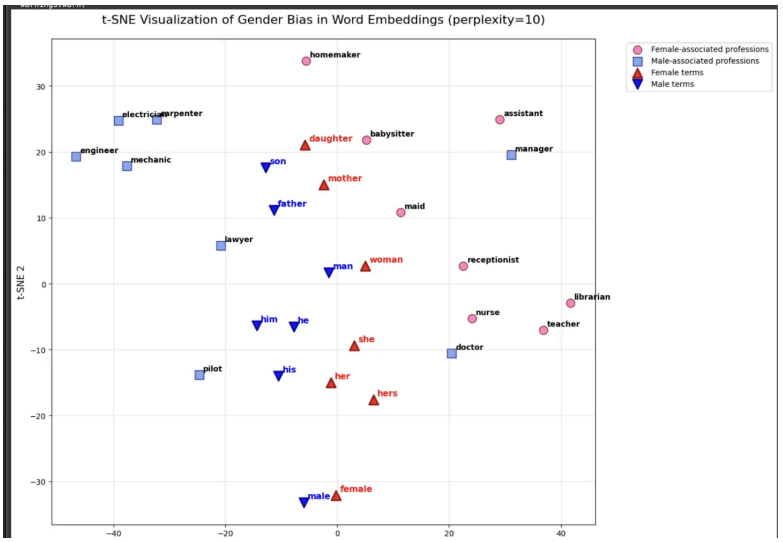
G. Heatmap



I plotted a heatmap of cosine similarities between profession terms and gendered attribute words (e.g., he, she, man, woman). This visualization helps reveal patterns of gender bias embedded in word representations. Using cosine similarity between profession terms and gendered words, I observed the following:

- Strong Female Bias: Professions like nurse, babysitter, homemaker, and maid show high similarity with female terms (e.g., woman, mother), indicating strong stereotypical associations.
- Weaker Male Bias: Male professions (e.g., engineer, carpenter, mechanic) show positive but less pronounced similarity with male terms. Some, like managers, appear neutral.
- Surprising Patterns: Doctor, often seen as male-coded, shows relatively high similarity with woman, suggesting evolving associations or dataset diversity.

H. T-SNE Visualization



To further explore bias, I used T-SNE plot to project high dimensional word embeddings into 2D space. This allows us to visually assess how professions and gendered terms cluster semantically.

- Professional Clustering along gender lines:
    - Female-associated professions cluster around female terms like mother, she, woman.
    - Male-associated professions form a separate cluster near male terms such as he, man, father.
- Care Work Cluster
    - Homemaker appears most isolated from male.
    - Babysitter is tightly aligned with daughter and mother.
    - Nurse, maid, and receptionist form a cohesive care/service-oriented group.
- Technical Profession Cluster (Top-Left)
    - Engineer, mechanic, electrician, and carpenter form a distinct cluster, away from female terms
    - Lawyer sits between clusters, reflecting a more neutral association

I. WEAT metrics

Metric	Value
WEAT Test Stat	1.4167
Effect Size	1.7208
p-value	0.0003

Target Set X = Stereotypically associated female professions
Target Set Y = Stereotypically associated male professions
Attribute set A = Female related terms
Attribute set B = Male related terms

- Test Statistic = 1.4167
    - Measures how much more strongly female professions (X) are associated with female terms (A) compared to male professions (Y).

        - A positive value indicates that X (female professions) are more semantically aligned with A (female terms) than Y is.
    - Effect Size = 1.7208
        - Standardizes the difference between the two groups' associations, in terms of standard deviation.
        - A value of 1.72 implies a very strong bias, meaning the profession groups are clearly separated in their gender associations
    - p-value = 0.0003
        - Calculated through permutations , this tells us the likelihood that such a strong bias could happen by chance.
        - .Very small p-value (< 0.001) means the bias is statistically significant and not random.

These results confirm that the Word2Vec embeddings carry clear gender stereotypes:

- Female professions are significantly closer to female terms in the embedding space.
- Male professions are relatively distant from female terms and more aligned with male terms.
- The effect is strong, consistent, and statistically significant

.

5.1.3 Limitations & Future Work

1. Limited Vocabulary Scope
   The study used a hand-picked list of professions and gender terms. This may not capture the full range of occupational or cultural bias present in the embedding space.
2. Model Dependency
   Analysis was restricted to Word2Vec. Other static embedding models (like GloVe, fastText) may exhibit different patterns or strengths of bias.
3. Lack of Contextual Sensitivity
   Word2Vec produces a single vector per word, ignoring context. This flattens out nuanced meanings (e.g., "nurse" as a verb vs noun) and may miss subtler bias patterns.

5.2 Evaluate Harmful Associations in Contextual Word Embeddings

- I used BERT embeddings (base uncased variant) to examine contextual bias, specifically gender and racial bias, by analyzing sentence pairs from the CrowS-Pairs dataset, a curated benchmark that contains matched sentence templates exhibiting social bias across several axes.
- The evaluation involves pairwise comparison of sentence likelihoods. For each biased/unbiased (or stereotypical/anti-stereotypical) pair, compute the model's preference using Pseudo Log-Likelihood (PLL) scores, which gives a scalar measure of how "natural" or "plausible" the model finds a sentence.
- Pseudo Log Likelihood:
    - The PLL score is computed by masking each token (excluding special tokens) one at a time in a sentence, predicting the masked word using the model, and summing the log-probabilities of the original tokens. This reflects the model's token-wise confidence in the full sentence.
    - A higher PLL for a stereotypical sentence compared to its anti-stereotypical counterpart suggests model bias—indicating that the model finds biased language more "natural".
    - Compared to static embedding evaluations (like cosine similarity between gendered word pairs), this approach captures bias in context, revealing not just associations between words, but how those associations play out across full sentence structures and diverse syntactic settings.
- I wrote a function to compute the Stereotype Preference Rate (SPR) using the CrowS-Pairs dataset. It checks how often BERT assigns a higher pseudo log-likelihood to the biased (stereotypical) sentence over the unbiased one, giving a clear signal of the model's bias.
- I also calculated a confidence score for each pair, which shows how strongly the model favors one sentence over the other. A high SPR with strong confidence suggests that BERT consistently leans toward biased language.
- Results:
    - Disability and Sexual Orientation Show Strong Bias. BERT showed high stereotype preference for disability (76.67%) and sexual-orientation (77.38%) with very high average confidence scores (0.75+). This suggests that the model

is consistently biased in favor of stereotypical completions in these domains, possibly due to disproportionate or skewed representations in pretraining data.
    - Gender and Nationality Biases Are Subtle but Present. For gender (49.2%) and nationality (44.6%), the scores hover around randomness, and average confidence is low (~0.47–0.48). This could mean the model isn't as overtly biased in these areas, or that the biases are more context-dependent or diffuse, escaping consistent pattern capture by log-likelihood comparisons.
    - Religion (69.5%) and physical appearance (61.9%) show moderately high bias scores, indicating stereotypical associations are preferred more often than not. Their confidence scores are also relatively high (0.65+), implying that these associations are strongly encoded in the model's predictions likely a reflection of entrenched social and media biases present in its training data.
- To go a step beyond just measuring stereotype preference rates, I developed a custom method to analyze which specific words within a sentence contribute most to the model's biased behavior. This involved creatively extending the pseudo-log-likelihood scoring approach.
- For each word in the sentence, I removed it and recalculated the score, measuring the change to determine that word's influence on the overall bias. This technique allowed me to isolate and quantify individual token contributions, offering a more granular lens into how BERT forms biased predictions.
- <u>Results:</u>
    - When comparing the two sentences, the word "woman" contributed far less to the bias score (4.34) than "man" did in the alternate sentence (6.60), suggesting that the model treats "man" as more informative or unexpected in this context. This subtle difference reflects how gendered terms impact the model's scoring asymmetrically, reinforcing gender stereotypes implicitly.

<u>Limitations</u>

- CrowS-Pairs Limitations: The pseudo-log-likelihood method only handles binary stereotype comparisons (e.g., male vs. female) and is currently restricted to English, leaving out multilingual bias analysis. It also implicitly assumes stereotypes are binary, which doesn't capture the full range of social identities or intersectionality.
- Sentence Fluency vs. True Bias: The PLL scores often conflate bias with fluency. A sentence might get a higher score just because it sounds more natural or familiar (due

to frequency in training data), not necessarily because it's more biased. This weakens the precision of the bias signal, especially in nuanced or low-frequency contexts.

- Word Contribution Analysis Caveats: My creative word removal method to trace bias-contributing tokens has its own trade-offs. Removing a word can break grammar or meaning, which BERT might penalize irrespective of the word's role in bias. Also, it treats each word's effect in isolation, ignoring compound effects or syntactic dependencies.
- I'd like to test models like RoBERTa, DistilBERT, or BERT-large, as their architecture, training objectives, or data size might yield different bias patterns. This would help gauge how robust or model-specific the bias findings are.

References

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, Hervé Jégou Word Translation Without Parallel Data

Aylin Caliskan, Joanna J. Bryson, Arvind Narayanan *Semantics derived automatically from language corpora contain human-like biases*