Semantic Representation and Word Analogies using Word2Vec

Responsible AI (AI 354) - Lab Assignment 5

Devesh Singh Chauhan
Roll No: I23MA002
M.Sc. Mathematics, SVNIT Surat

February 25, 2026

1 Introduction

This lab investigates continuous vector representations of words using the pre-trained Google News Word2Vec model. By projecting words into a high-dimensional vector space ($\mathbb{R}^{300}$), we can capture semantic meanings and syntactic relationships. The assignment explores two main applications: constructing sentence-level embeddings from word vectors for sentiment data (Amazon Reviews) and evaluating semantic relationships through algebraic word analogies.

2 Q1: Text Preprocessing & Sentence Representation

2.1 Methodology

To convert raw text into a format suitable for vectorization, a standard NLP pipeline was applied to a subset of 1,000 Amazon reviews:

1. **Case Folding:** All text was converted to lowercase to maintain uniformity (e.g., "Apple" and "apple" map to the same vector).
2. **Punctuation & Number Removal:** Non-alphabetic characters were stripped using regular expressions, as they generally carry little semantic weight in standard Word2Vec models.
3. **Tokenization:** Sentences were split into discrete word tokens.
4. **Stopword Removal:** Frequent but semantically hollow words (e.g., "the", "is", "in") were removed using the NLTK English stopword list.

Example Execution:

- *Original:* "This sound track was beautiful! It paints the senery in your mind so well I would recomend it even t..."
- *Tokens:* `['sound', 'track', 'beautiful', 'paints', 'senery', 'mind', 'well', 'would', 'recomend', 'even']`

2.2 Sentence Representation via Mean Pooling

Word2Vec provides embeddings for individual words. To represent an entire sentence, we compute the centroid (mean) of its constituent word vectors. Let a sentence S consist of N valid tokens $\{w_1, w_2, \ldots, w_N\}$. The sentence vector $V_S \in \mathbb{R}^{300}$ is:

$$V_S = \frac{1}{N} \sum_{i=1}^{N} \vec{w}_i$$

2.3 PCA Visualization Analysis

To visualize the 300-dimensional sentence representations, Principal Component Analysis (PCA) was utilized to project the vectors onto a 2D plane ($\mathbb{R}^{300} \rightarrow \mathbb{R}^2$).

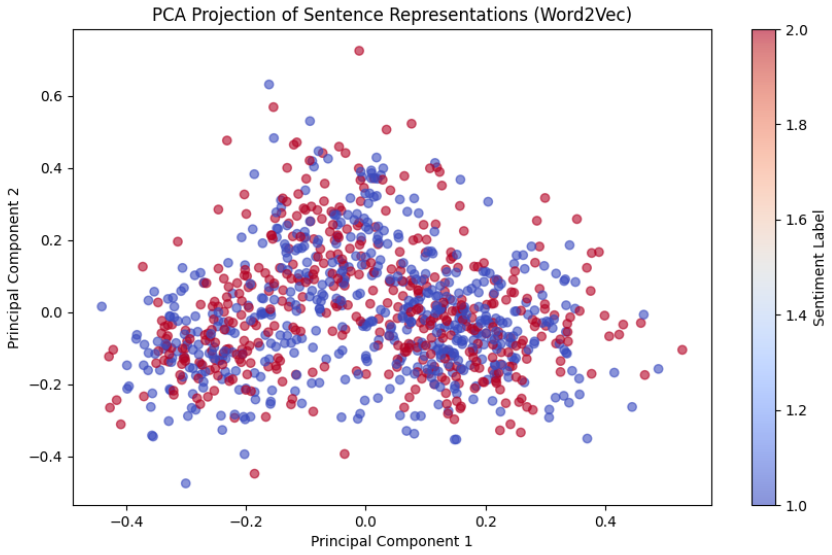

Figure 1: PCA projection of sentence vectors, colored by sentiment label.

Observation: The plot reveals a dense, overlapping cluster of sentence vectors without distinct boundaries between positive and negative sentiments. This highlights a fundamental limitation of *mean pooling*: it treats a sentence as a "bag of words," completely ignoring word order and context. For example, "not good" and "good" may end up close together because the semantic weight of "good" dominates, making basic Word2Vec averaging suboptimal for complex sentiment analysis compared to sequential models like LSTMs.

3 Q2: Word Analogies and Vector Space Mathematics

Word2Vec models linear relationships between words. The analogy task asks: *"A is to B as C is to D"*. Mathematically, we solve for vector w_d such that:

$$\vec{w}_b - \vec{w}_a \approx \vec{w}_d - \vec{w}_c \implies \vec{w}_d \approx \vec{w}_b - \vec{w}_a + \vec{w}_c$$

We find the word in the vocabulary whose vector has the highest cosine similarity to $\vec{w}_d$.

3.1 Specific Analogy Tests

Analogy A: `good - bad + excellent = ?`

- *Expected Logic:* The relationship is "degree of quality." If 'excellent' is an extreme version of 'good', we want the extreme version of 'bad'.

- *Model Predictions:* `['terrific', 'superb', 'fantastic']`
- *Analysis:* The model failed to return negative extremes (like "terrible"). Instead, the strong positive magnitude of "excellent" combined with "good" overpowered the subtraction of "bad," pushing the resultant vector deep into the cluster of highly positive adjectives.

Analogy B: `delivery - late + fast = ?`

- *Expected Logic:* The opposite of a late delivery is a fast delivery/shipping.
- *Model Predictions:* `['delievery', 'twitch_muscles', 'Delivery']`
- *Analysis:* The model returned variations of the word "delivery" (including a typo, 'delievery'). This occurs because the semantic distance between "late" and "fast" did not create a meaningful enough translation vector in the specific context of the Google News corpus to shift away from the dense neighborhood surrounding the word "delivery".

3.2 Search Space Minimization Strategy

The Google Word2Vec model contains 3,000,000 vocabulary words. Finding the most similar word by calculating cosine similarity against 3 million vectors for every analogy question is computationally expensive. We employed two optimization techniques:

1. **Vocabulary Restriction (`restrict_vocab`):** By limiting the search to the top 300,000 most frequent words, we eliminate 90% of the search space. The excluded 2.7 million words are largely noise (typos, obscure names, rare acronyms), meaning accuracy is preserved while computation time is drastically reduced.
2. **L2 Normalization (`fill_norms`):** Standard cosine similarity requires dividing the dot product by the magnitudes of both vectors:

$$\cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

   By pre-calculating the L2 norms ($\|A\|$) of all vectors in the matrix, the algorithm can bypass the expensive division step during the search phase, relying solely on highly optimized matrix dot-product operations.

3.3 Analogy Dataset Evaluation

The optimized model was evaluated on the benchmark `questions-words.txt` dataset.

Metric	**Result**
Optimization Strategy	Top 300,000 words + L2 Normalization
Overall Accuracy	74.01%
Execution Time	228.28 seconds

Table 1: Performance on the Word Analogy Benchmark.

Conclusion: An accuracy of 74.01% confirms that the Word2Vec model successfully captures complex syntactic and semantic regularities. The optimizations allowed the entire evaluation to complete in roughly 3.8 minutes, proving that restricting the vocabulary to dense, high-frequency regions efficiently filters out noise without sacrificing structural integrity.