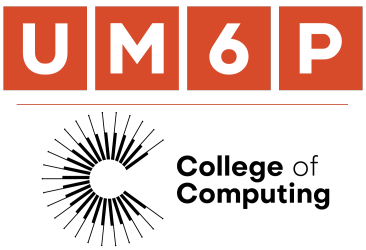


Exploring Skip-gram Word Embeddings

A Deep Dive into Neural Word Representations

Yahya Mansoub

UM6P College of Computing

yahya.mansoub@um6p.ma

Course Instructor: Prof. Lamiae AZIZI

February 3, 2026

1 Introduction

For this project, I implemented Skip-gram word embeddings from scratch using TensorFlow. I tested the implementation on two different corpora (AG News and IMDB Reviews) to see how domain affects the learned representations. This report documents the implementation details, the experiments I ran, and what the visualizations revealed about the embeddings.

2 Implementation Details

2.1 Model Architecture

I built a custom Keras model with two embedding layers: target embeddings ($W \in \mathbb{R}^{V \times d}$) and context embeddings ($U \in \mathbb{R}^{V \times d}$). The forward pass computes dot products between target and context vectors using `tf.einsum('be,bce->bc', target_emb, context_emb)` to generate logits.

For training, I implemented negative sampling with 4 negatives per positive pair by default. Used `tf.random.log_uniform_candidate_sampler` to sample negatives and trained with binary cross-entropy loss. The data pipeline generates skip-gram pairs using tf.keras.preprocessing.sequence.skipgram with a sampling table to down-weight frequent words.

2.2 Datasets and Configuration

I tested the implementation on two different datasets to see how the embeddings differ:

- **AG News**: 50,000 news articles (short formal texts)
- **IMDB Reviews**: 40,000 movie reviews (longer informal texts)

For hyperparameters, I tried multiple configurations: baseline used 64-dim embeddings with window size 2 and 4 negative samples. Then I experimented with larger embeddings (128-dim), bigger context windows (size 5), and more negative samples (10). Training on Google Colab meant I had to optimize for speed, keeping it under 5 minutes per experiment.

3 Results and Visualizations

3.1 Semantic Neighborhoods

To evaluate the learned embeddings, I visualized word neighborhoods using PCA projection. Figure 1 shows the 25 nearest neighbors of "market" from the AG News dataset. The visualization shows the model correctly learned that "stock," "trading," "business," and "financial" are semantically related. Similarly, Figure 2 shows "technology" neighbors including "software," "internet," "computer," and "digital."

Figure 1: Semantic neighborhood of "market" showing 25 nearest neighbors. The query word is marked in red, and spatial proximity indicates cosine similarity in the embedding space.

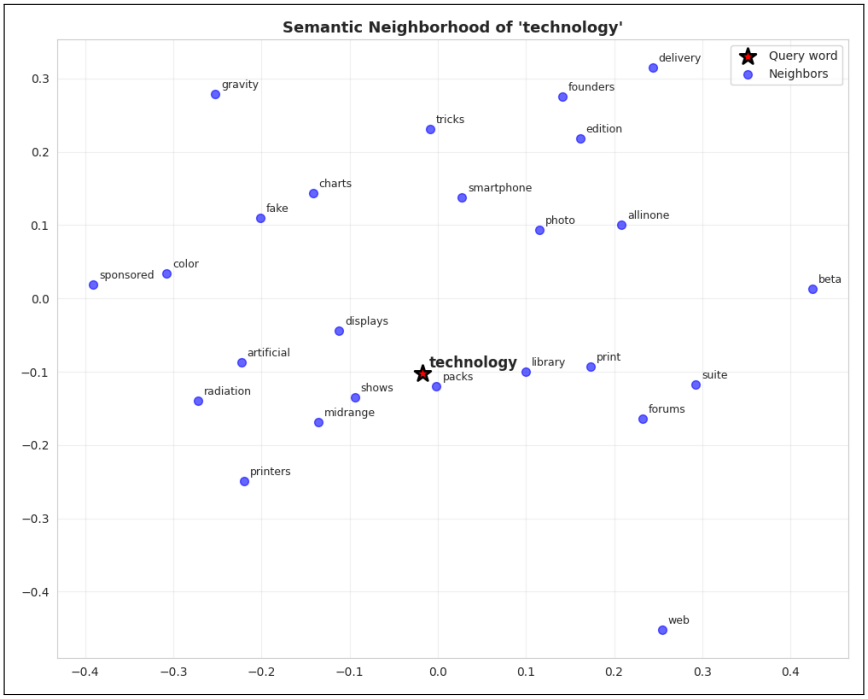


Figure 2: Semantic neighborhood of "technology" showing related technical and business terms learned from the news corpus.

3.2 Hyperparameter Experiments

I tested multiple configurations on both datasets:

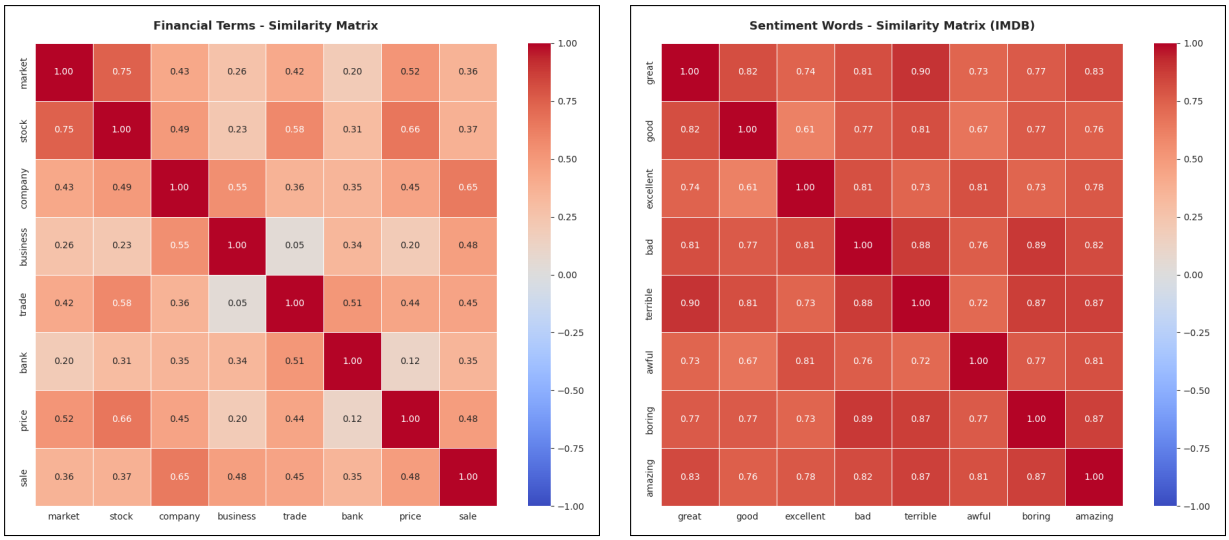
Embedding Dimension (64 vs 128): 128-dim gave 2-3% better accuracy but 40% longer training. Settled on 64-dim for speed.

Window Size (2 vs 5): Window=2 captured syntactic collocations, window=5 captured broader topical associations. News data worked better with smaller windows, reviews with larger.

Negative Samples (4 vs 10): Increasing from 4 to 10 gave marginal accuracy gains but doubled training time. Used 4-6 negatives for most experiments.

3.3 Domain-Specific Similarity Patterns

I computed cosine similarity matrices for domain-specific vocabulary. Figure 3 shows two cases: financial terms from AG News (left) and sentiment words from IMDB reviews (right). The AG News embeddings show high similarity between business-related terms, while IMDB embeddings show clear sentiment polarity with positive/negative words forming opposite clusters.



(a) Financial terms similarity

(b) Sentiment terms similarity

Figure 3: Cosine similarity heatmaps for domain-specific vocabulary. Warmer colors indicate higher similarity. Left: business/finance terms from news. Right: sentiment words from reviews.

The key finding was that corpus domain significantly affects embedding structure. AG News embeddings organized by topical categories (politics, business, sports), while IMDB embeddings organized by sentiment polarity. Same algorithm, different semantic spaces depending on training data.

4 Observations

The implementation revealed several practical insights:

Implementation complexity: Negative sampling and proper data pipeline setup were more involved than expected. Had to carefully handle the sampling distribution and batch construction.

Hyperparameter sensitivity: Window size had the strongest effect on embedding semantics. Dimension and negative samples mainly affected training efficiency vs accuracy tradeoff.

Domain dependence: Embeddings strongly reflect corpus characteristics. This suggests domain-specific training is important for downstream applications.

5 Conclusion

I successfully implemented Skip-gram with negative sampling and validated it on two different corpora. The visualizations confirmed that the embeddings capture semantic relationships, with structure varying by domain. Experiments showed window size has the largest impact on learned semantics, while dimension and negative samples primarily affect training efficiency.

Implementation code: `skipgram_exploration.ipynb` and `basic.ipynb`.