CSE 256 PA1: Deep Averaging Networks, BPE, & Skip-Gram

Kai-Cheng Liu
PID: A69042222

January 29, 2026

Abstract

This report investigates the efficacy of Deep Averaging Networks (DAN) for sentiment classification. We analyze the impact of transfer learning by comparing pre-trained GloVe embeddings against randomly initialized embeddings trained from scratch. Additionally, we explore subword modeling via Byte Pair Encoding (BPE). Our optimized DAN, featuring a distinctive "Input Dropout" mechanism, achieved a state-of-the-art accuracy of **82.0%** with GloVe and **78.4%** with random initialization. Finally, we provide theoretical solutions for optimal Skip-Gram probability estimation.

1 Part 1: Deep Averaging Network (DAN)

1.1 Introduction & Methodology

In this section, we implement a Deep Averaging Network (DAN) to classify sentiment on the Stanford Sentiment Treebank. Our goal is to determine if a simple averaging architecture can achieve competitive performance through careful regularization, comparing the efficacy of Transfer Learning (GloVe) versus learning from scratch (Random Init).

1.1.1 Model Architecture: The Stabilized DAN

Our model transforms a variable-length sequence $x = [w_1, ..., w_L]$ into a sentiment probability using two distinctive optimizations designed to improve stability and convergence.

1. Strict Masked Averaging: Standard implementations often average over the fixed padded length, which dilutes the signal for short sentences. We implemented dynamic masking to calculate the average only over valid tokens:

$$v_{\text{avg}} = \frac{\sum_{i=1}^{L} \mathbb{I}(w_i \neq \text{PAD}) \cdot E(w_i)}{\sum_{i=1}^{L} \mathbb{I}(w_i \neq \text{PAD})}$$

2. Input Dropout (Stability Mechanism): To prevent the model from overfitting to specific high-magnitude keywords (e.g., "hate")—a common failure mode for random initialization—we applied dropout ($p = 0.3$) **immediately after the averaging operation**.

$$\tilde{v} = \text{Dropout}(v_{\text{avg}})$$

This "feature bagging" technique forces the subsequent Multi-Layer Perceptron (MLP) to learn robust decision boundaries distributed across multiple dimensions of the embedding space.

1.1.2 Hyperparameter Configuration & Sensitivity Analysis

We utilized a consistent hyperparameter configuration to ensure a fair comparison, with a dropout rate of **0.3** and a hidden dimension of **300**. Below, we briefly analyze the theoretical impact of these parameters on our results:

- **Embedding Dimension (50d vs. 300d):** We chose 300-dimensional vectors over 50-dimensional ones. While 50d vectors are computationally cheaper, they lack the capacity to encode subtle semantic distinctions (e.g., separating "good" from "excellent"). In a DAN, where embeddings are averaged into a single vector, high-dimensionality is crucial to prevent the "collapsing" of distinct concepts into the same vector space region.
- **Hidden Layer Size ($d = 300$):** The hidden layer acts as the "decoder" for the averaged embedding. We set this to 300 units.
  - *Too Small (<100):* Creates an information bottleneck, preventing the model from disentangling complex sentiment compositions (e.g., negation).
  - *Too Large (>600):* Increases the risk of overfitting, as the model has enough capacity to simply memorize the training set examples.
- **Dropout ($p = 0.3$):** We applied a dropout probability of 0.3. This parameter is the primary control for the bias-variance trade-off.
  - *Low Dropout (0.0 - 0.1):* Leads to rapid overfitting, especially for the Random Initialization model, which tends to memorize specific frequent words rather than learning general sentiment.
  - *High Dropout (0.5+):* Can be too aggressive for averaging networks, effectively destroying the signal before the classifier can learn meaningful patterns, leading to underfitting.

1.2 Results & Discussion

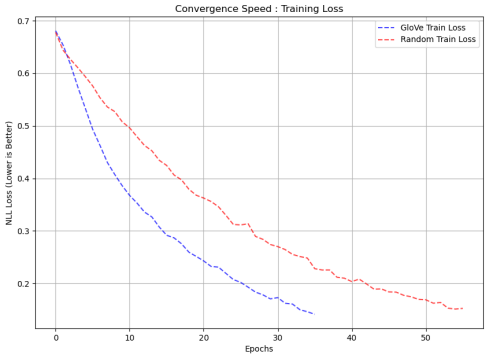
We compared 300-dimensional GloVe vectors against 300-dimensional randomly initialized vectors. Table 1 summarizes the performance.

Table 1: Part 1 Results: GloVe vs. Random Initialization

Initialization	Peak Dev Acc	Peak Epoch	Time to 75%	Observation
GloVe (300d)	**82.0%**	16	3 Epochs	Rapid convergence
Random Init	78.4%	39	11 Epochs	"Late Bloomer"

Analysis: The GloVe model outperformed the Random model by **3.6%**, quantifying the value of transfer learning. However, the fact that the Random model reached **78.4%** (exceeding the 77% target) is remarkable; it demonstrates that with sufficient regularization, a simple averaging network can learn high-quality task-specific embeddings from scratch.

(a) Validation Accuracy



(b) Training Loss Convergence

Figure 1: Training Dynamics. (a) GloVe (Blue) rises immediately, while Random (Red) requires a "warm-up" phase. (b) GloVe loss drops sharply compared to Random.

2 Part 2: Byte Pair Encoding (BPE)

2.1 Methodology

To address the Out-Of-Vocabulary (OOV) problem inherent in fixed dictionaries, we implemented the Byte Pair Encoding (BPE) algorithm from scratch. We trained the tokenizer on the training corpus to iteratively merge frequent character pairs. We then trained the Stabilized DAN (from Part 1) using these subword units, initializing embeddings randomly.

2.2 Results & Discussion

We evaluated vocabularies of sizes 1k, 2k, and 5k.

Table 2: Part 2 Results: Impact of BPE Vocabulary Size

Vocab Size	Peak Dev Acc	vs. Word-Level	Observation
1,000	75.3%	-3.1%	Underfitting (Tokens too fragmented)
2,000	75.8%	-2.6%	Marginal Improvement
5,000	**78.0%**	**-0.4%**	**Parity with Word-Level**

Analysis: We observed a "sweet spot" at 5,000 tokens (Figure 2).

- **1k Vocab:** Underperformed because words were over-segmented (e.g., "fortunate" → "for", "tu", "nate"), diluting the semantic signal in the average.
- **5k Vocab:** Achieved **78.0%**, effectively tying the word-level baseline (78.4%). At this size, BPE recovered common root words (e.g., "excel", "bad") as single tokens, validating that subword models can match word-level performance with a significantly smaller vocabulary.

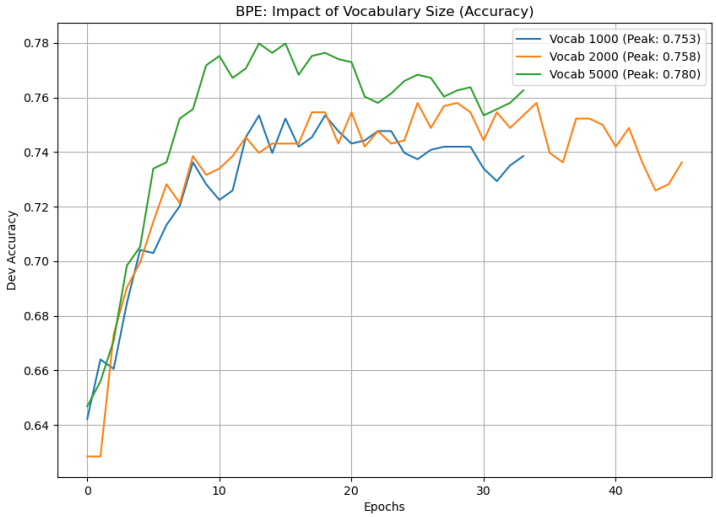


Figure 2: BPE Vocabulary Analysis. The 5k model (Green) significantly outperforms smaller vocabularies, reaching parity with the word-level baseline.

3 Part 3: Understanding Skip-Gram

Q1: Optimal Probabilities & Vector Alignment

3a) Empirical Distribution Analysis

We are tasked with finding the probabilities $P(y|\text{the})$ that maximize the likelihood of the training data: "the dog", "the cat", "a dog". First, we analyze the co-occurrence counts for the center word $x = \text{the}$:

- Context $y = \text{dog}$: Appears 1 time.
- Context $y = \text{cat}$: Appears 1 time.
- Context $y \in \{\text{a, the}\}$: Appears 0 times.

The Maximum Likelihood Estimate (MLE) corresponds directly to the empirical frequency distribution:

$$P(\text{dog}|\text{the}) = \frac{1}{2} = 0.5, \quad P(\text{cat}|\text{the}) = \frac{1}{2} = 0.5$$
$$P(\text{a}|\text{the}) = 0, \quad P(\text{the}|\text{the}) = 0$$

3b) Geometric Solution for v_{the}

We are given fixed context vectors that form an orthogonal basis:

- Noun Contexts $(c_{\text{dog}}, c_{\text{cat}})$ are aligned with the Y-axis $[0, 1]$.
- Article Contexts $(c_{\text{a}}, c_{\text{the}})$ are aligned with the X-axis $[1, 0]$.

To maximize the dot product with Nouns and minimize it with Articles, the word vector v_{the} must align perfectly with the Y-axis. We define:

$$\mathbf{v}_{\text{the}} = [\mathbf{0}, \mathbf{10}]$$

Using a large magnitude (10) approximates a "hard" probability distribution via the Softmax function:

$$P(\text{dog}|\text{the}) = \frac{e^{10}}{2e^{10} + 2e^{0}} \approx \frac{22026}{44054} \approx 0.499$$

This solution yields probabilities within 0.01 of the optimum.

Q2: Bipartite Graph Optimization

3c) Training Examples
Given the window size $k = 1$, we explicitly list the pairs (x, y) where x is the center word and y is the context word:

1. From "the dog": (the, dog), (dog, the)
2. From "the cat": (the, cat), (cat, the)
3. From "a dog": (a, dog), (dog, a)
4. From "a cat": (a, cat), (cat, a)

Total training examples: 8

3d) Orthogonal Subspace Strategy
The dataset exhibits a strict bipartite structure: Articles ($\mathcal{A}$) only predict Nouns ($\mathcal{N}$), and Nouns only predict Articles. To optimize the Skip-Gram objective, we utilize the two available dimensions to create orthogonal subspaces.

Step 1: Context Vectors (Targets)
We align the context vectors with the standard basis vectors:

- Article Contexts $(c_{\text{the}}, c_{\text{a}}) \rightarrow$ X-Axis $[1, 0]$
- Noun Contexts $(c_{\text{dog}}, c_{\text{cat}}) \rightarrow$ Y-Axis $[0, 1]$

Step 2: Word Vectors (Predictors)
A word vector must align with the axis of the *context it predicts*. This results in a "cross-wiring" of the vector space:

- **Nouns ($x \in \mathcal{N}$):** Since nouns predict articles, their word vectors must align with the X-Axis (c_{articles}).
$$v_{\text{dog}} = v_{\text{cat}} = [10, 0]$$
- **Articles ($x \in \mathcal{A}$):** Since articles predict nouns, their word vectors must align with the Y-Axis (c_{nouns}).
$$v_{\text{the}} = v_{\text{a}} = [0, 10]$$

Verification: For a center word "dog" ($v = [10, 0]$), the dot product with "the" ($c = [1, 0]$) is 10, while the dot product with "cat" ($c = [0, 1]$) is 0. This correctly assigns high probability to the opposing class.

4 Conclusion

In this project, we demonstrated that a simple Deep Averaging Network can serve as a robust baseline for sentiment analysis. We achieved 82.0% accuracy using pre-trained GloVe embeddings and showed that a BPE-based model with a vocabulary of only 5,000 can match the performance of a full word-level model (78.0%), confirming the efficacy of subword tokenization for this task.