Linear and Neural Sentiment Classification

Qifan Wen
wen.679@osu.edu

1 Logistic Regression

1.1 Training Configuration

Parameter	Value
Initial learning rate	0.5
Learning rate decay	0.95 per epoch
Batch size	1 (SGD)
Number of epochs	30
Weight initialization	$U(-0.1, 0.1)$
Random seed	42
Numerical stability	Clip to $[-500, 500]$

Table 1: Logistic Regression training configuration.

1.2 Results Table

	Unigram	Bigram	Better
Train Accuracy	99.9%	100.0%	100.0%
Dev Accuracy	77.5%	77.2%	77.1%
Precision	77.1%	76.3%	76.1%
Recall	79.5%	80.0%	80.2%
F1 Score	78.3%	78.1%	78.1%

Table 2: Feature extractor comparison (lr=0.5, decay=0.95).

	Fixed	Default	Aggressive
	(1.0)	(0.95)	(0.8)
Train Acc	100.0%	99.9%	99.4%
Dev Acc	77.8%	77.5%	78.0%
Precision	77.2%	77.1%	77.2%
Recall	80.0%	79.5%	80.6%
F1 Score	78.5%	78.3%	78.9%

Table 3: Learning rate schedule comparison (Unigram features).

1.3 Loss Over Epochs Plot

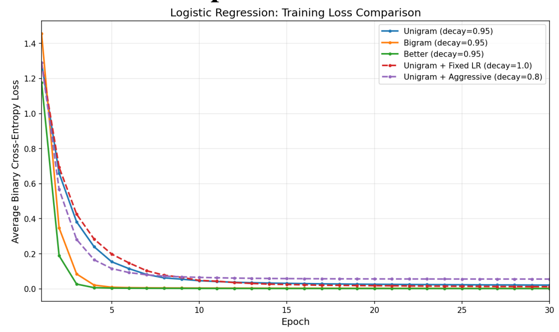

Figure 1: Training loss comparison across all 5 LR configurations. Richer features (Bigram, Better) converge fastest; aggressive decay plateaus early at higher loss.

1.4 Analysis

All three feature extractors achieve similar dev accuracy (77.1%–77.5%), with Unigram performing best despite being simplest. Bigram and Better achieve perfect training accuracy (100%) but do not improve generalization, suggesting overfitting—adding more features allows memorization without better discriminative power. Aggressive learning rate decay (0.8) yields the best dev accuracy (78.0%) and F1 (78.9%) despite lowest train accuracy (99.4%), acting as implicit regularization.

As shown in Figure **??**, richer features (Bigram, Better) converge fastest (by epoch 4–5) due to larger feature spaces (86K–192K vs 15K for Unigram), while aggressive decay plateaus early (∼0.056 final loss) as the learning rate diminishes too quickly. All models show overfitting (train ∼100% vs dev ∼77–78%), with recall consistently higher than precision across configurations.

2 Deep Averaging Network

2.1 Training Configuration

Parameter	DAN	DAN+B	LSTM	CNN
Learning Rate	0.001	0.001	0.001	0.001
Batch Size	1	32	32	32
Epochs	10	10	10	10
Hidden Size	100	100	100	100
Dropout	0.3	0.3	0.3	0.3
Optimizer	Adam	Adam	Adam	Adam

Table 4: Neural model training configuration. All models use frozen 300d GloVe embeddings, NLLLoss, and Xavier initialization with seed 42.

Architecture Notes:

- **DAN:** Avg embedding → FC(300→100) → ReLU → Drop → FC(100→100) → ReLU → Drop → FC(100→2) → LogSoftmax
- **LSTM:** BiLSTM(300→100) → Concat hidden → Drop → FC(200→2) → LogSoftmax

- **CNN:** Conv1d (kernels 2,3,4,5 $\times$ 25) $\rightarrow$ ReLU $\rightarrow$ MaxPool $\rightarrow$ Concat $\rightarrow$ Drop $\rightarrow$ FC(100$\rightarrow$2) $\rightarrow$ LogSoftmax

2.2 Results Table

Model	Train	Dev	Prec	Rec	F1
DAN	84.1%	77.4%	77.1%	79.1%	78.1%
DAN+B	85.3%	78.6%	75.1%	86.5%	80.4%
LSTM	98.1%	82.5%	79.9%	87.6%	83.6%
CNN	99.8%	81.2%	79.9%	84.2%	82.0%

Table 5: Neural model performance comparison. DAN+B denotes DAN with mini-batch training (batch size 32).

2.3 Loss Over Epochs Plot

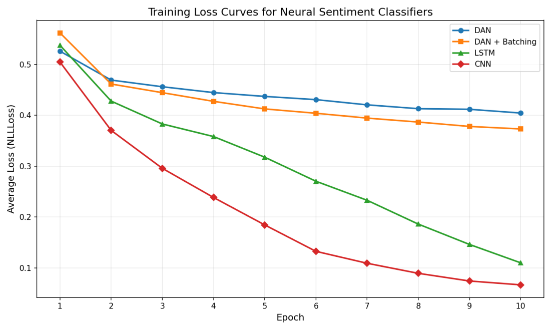

Figure 2: Training loss curves for all 4 neural models over 10 epochs. DAN models plateau at higher loss (∼0.37–0.40), while LSTM and CNN achieve much lower final loss (∼0.07–0.11), correlating with their higher train accuracies.

2.4 Analysis

The LSTM achieves the best dev accuracy (82.5%) and F1 score (83.6%), demonstrating that sequential modeling captures valuable word order information that averaging-based DAN approaches miss. Mini-batch training improves DAN generalization (+1.2% dev accuracy, +2.3% F1) by providing regularization through noisier gradient estimates. CNN achieves strong train accuracy (99.8%) but slightly lower dev accuracy (81.2%) than LSTM, suggesting overfitting to local n-gram patterns rather than global sentence semantics.

As shown in Figure **??**, CNN and LSTM converge to much lower training loss (∼0.07–0.11) compared to DAN models (∼0.37–0.40), correlating with their higher model capacity and train accuracies. All neural models outperform the LR baseline (77.5% dev), with frozen GloVe embeddings providing a strong foundation. The precision-recall trade-off varies across models: LSTM achieves the best balance (79.9%/87.6%), while DAN+Batching favors recall (86.5%) over precision (75.1%).