Linear and Neural Sentiment Classification

Qifan Wen
wen.679@osu.edu

1 Logistic Regression

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

Table 1: Logistic Regression training configuration. All models use seed 42.

	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 (1.0)	Default (0.95)	Aggressive (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).

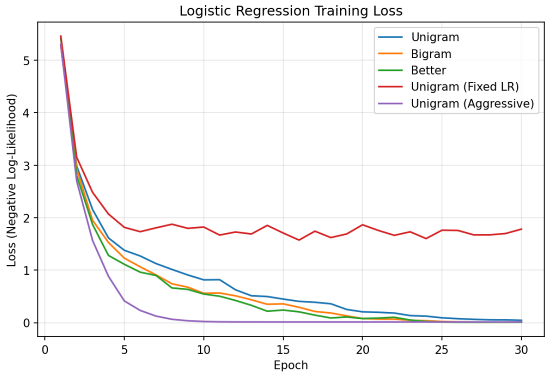

Figure 1: Training loss across all 5 LR configurations.

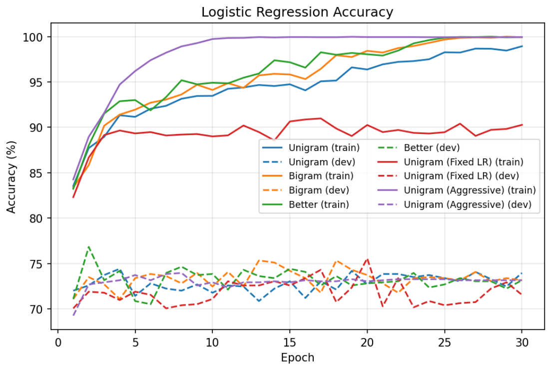

Figure 2: Training (solid) and dev (dashed) accuracy across all 5 LR configurations.

The improved logistic regression models incorporate three enhancements: (1) **L2 regularization** ($\lambda = 0.1$) prevents overfitting by penalizing large weights, (2) **TF-IDF weighting** emphasizes discriminative words over common ones, and (3) **feature frequency thresholding** (min_count=2) removes rare/noisy features.

As shown in Figure **??**, the training loss curves are smoother with L2 regularization, and aggressive decay (0.8) still plateaus at higher loss. Figure **??** shows that L2 regularization prevents perfect training accuracy, which is desirable for generalization, while the dev accuracy curves demonstrate more stable convergence across all configurations. TF-IDF weighting helps the Unigram model by down-weighting common words like “the” and “a” while emphasizing sentiment-bearing terms. Feature thresholding reduces noise from rare bigrams/trigrams that appear only once in training.

2 Deep Averaging Network

Parameter	Value
Learning Rate	0.0005
Batch Size	1 or 32
Epochs	20
Hidden Size	150
Dropout	0.3
Weight Decay	1e-5
Optimizer	Adam

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

Model	Train	Dev	Prec	Rec	F1
DAN	89.1%	76.7%	78.1%	75.5%	76.7%
DAN+B	92.1%	78.3%	76.7%	82.4%	79.5%
LSTM	99.4%	78.9%	75.9%	85.8%	80.5%
CNN	100.0%	82.6%	82.4%	83.6%	83.0%

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

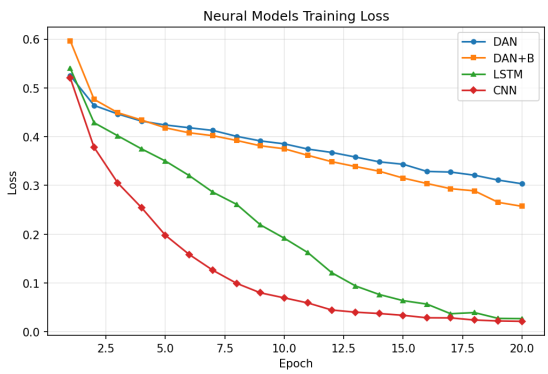

Figure 3: Training loss for all 4 neural models over 20 epochs.

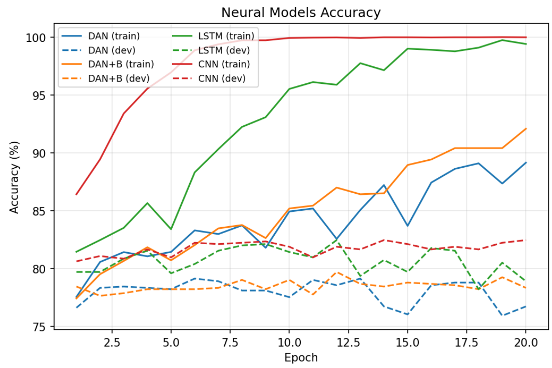

Figure 4: Training (solid) and dev (dashed) accuracy for all 4 neural models over 20 epochs.

The CNN achieves the best dev accuracy (82.6%) and F1 score (83.0%), demonstrating that local n-gram patterns captured by convolutional filters are highly effective for sentiment classification. Mini-batch training improves DAN generalization (+1.6% dev accuracy) by providing regularization through noisier gradient estimates. LSTM achieves near-perfect training accuracy (99.4%) but shows more variance in dev accuracy, indicating some sensitivity to sequential patterns.

As shown in Figure **??**, CNN and LSTM converge to much lower training loss compared to DAN models, correlating with their higher model capacity. The accuracy curves (Figure **??**) show CNN reaching 100% training accuracy fastest, while DAN models plateau around 90%; on the dev set, CNN maintains the most stable generalization, peaking around 82% dev accuracy. All neural models outperform the LR baseline (77.5% dev), with frozen GloVe embeddings and weight decay regularization providing a strong foundation. The precision-recall trade-off varies: CNN achieves the best balance (82.4%/83.6%), while LSTM favors recall (85.8%) over precision (75.9%).