Linear and Neural Sentiment Classification

Qifan Wen
`wen.679@osu.edu`

1 Logistic Regression

Parameter	**Value**
Initial learning rate	0.2
Learning rate decay	0.95 per epoch
Batch size	32 (mini-batch SGD)
Number of epochs	30 (max)
L2 regularization (λ)	0.1
Min feature count	2
Early stopping patience	4 epochs
Weight initialization	$U(-0.1, 0.1)$
Numerical stability	Clip to $[-500, 500]$

Table 1: Logistic Regression training configuration selected by random grid search (20 of 162 combinations, seed 42).

	Unigram	**Bigram**	**Better**
Train Accuracy	81.9%	82.4%	84.4%
Dev Accuracy	76.2%	76.7%	77.9%
Precision	78.6%	77.6%	79.3%
Recall	73.0%	76.4%	76.6%
F1 Score	75.7%	77.0%	77.9%

Table 2: Feature extractor comparison (lr=0.2, decay=0.95, grid-searched hyperparameters).

	Fixed	**Default**	**Aggressive**
	(1.0)	(0.95)	(0.8)
Train Acc	80.0%	81.9%	88.0%
Dev Acc	75.1%	76.2%	78.8%
Precision	72.6%	78.6%	78.5%
Recall	82.2%	73.0%	80.4%
F1 Score	77.1%	75.7%	79.4%

Table 3: Learning rate schedule comparison (Unigram features, grid-searched hyperparameters).

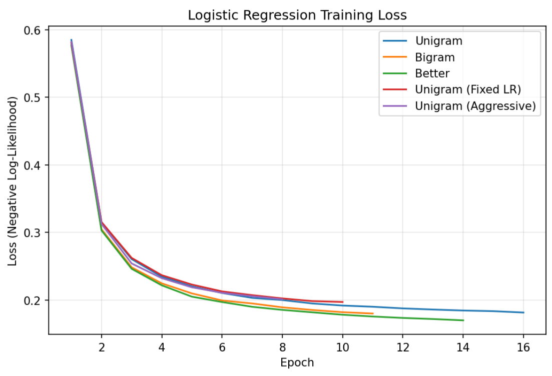

Figure 1: Training loss across all 5 LR configurations.

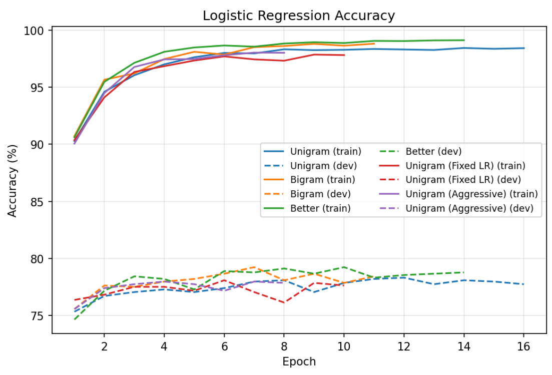

Figure 2: Training (solid) and dev (dashed) accuracy across all 5 LR configurations.

The logistic regression model incorporates five enhancements selected via random grid search (20 of 162 hyperparameter combinations). Mini-batch SGD with batch size 32 provides smoother gradient estimates than per-example updates. L2 regularization ($\lambda = 0.1$) prevents overfitting by penalizing large weights. Early stopping with patience of 4 epochs halts training when dev accuracy plateaus. TF-IDF weighting emphasizes discriminative words over common ones. Feature frequency thresholding (min_count = 2) removes rare and noisy features.

Two exploration tasks were implemented to evaluate how feature selection and learning rate decay affect model performance. Better feature selection significantly improves performance, while learning rate decay has only limited impact.

2 Deep Averaging Network

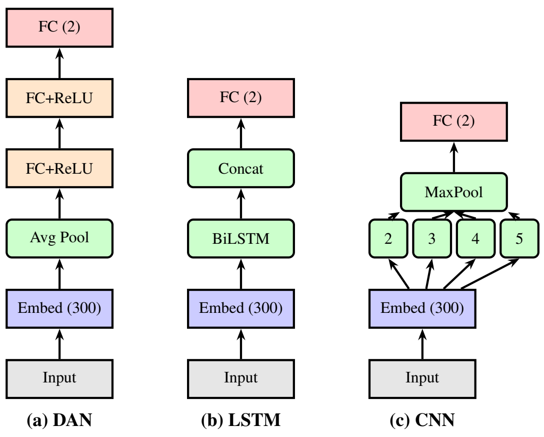

Figure 3: Neural architectures.

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).

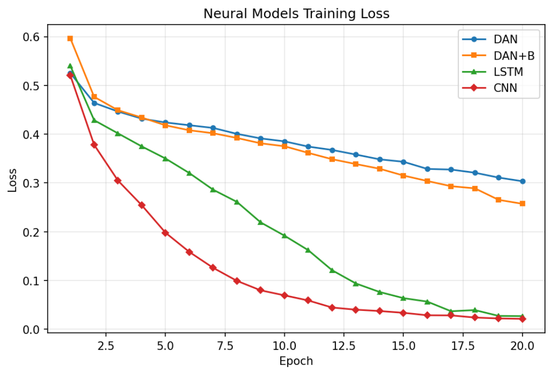

Figure 4: Training loss for all 4 neural models over 20 epochs.

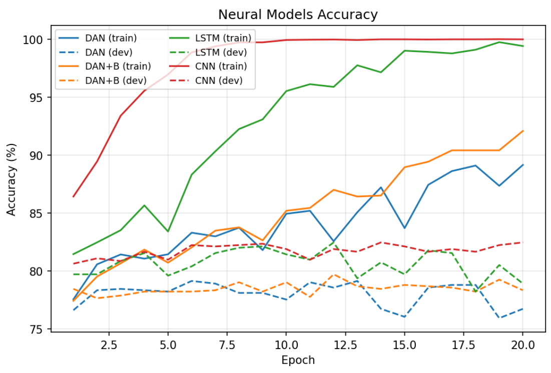

Figure 5: Training (solid) and dev (dashed) accuracy for all 4 neural models over 20 epochs.

Two exploration tasks were implemented. 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). LSTM achieves near-perfect training accuracy (99.4%) but shows more variance in dev accuracy.

As shown in Figure 4, CNN and LSTM converge to much lower training loss compared to DAN models, correlating with their higher model capacity. The accuracy curves (Figure 5) 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%).