Workshop 3: RNN & RLSTM

From-Scratch RNN, PyTorch RLSTM, and Text Classification on a Kaggle Dataset

Arman Golbidi

Course: BM20A6100 Advanced Data Analysis and Machine Learning

Period: 1 Sep 2025 – 12 Dec 2025

This report is written in LaTeX

Week 3 workshop

Contents

1 Introduction **3**

2 Dataset **3**

3 Methods **3**
3.1 Vanilla RNN (from-scratch concept) 3
3.2 RLSTM (PyTorch BiLSTM classifier) 4
3.3 Flowcharts (pipeline overview) 4

4 Experiments and Results **7**
4.1 Environment 7
4.2 Hyperparameter Tuning 7
4.3 Baseline RNN (PyTorch) — Behaviour 7
4.4 RLSTM (BiLSTM) — Results with Analysis 8
4.5 Final Metrics 11

5 Comparison: RNN vs. RLSTM **11**

6 Discussion **12**

A Selected Optuna Logs (abridged) **12**

1 Introduction

Scope. This workshop has two parts: (1) we implement a minimal recurrent neural network (RNN) *from scratch* in `NumPy` to understand sequence state updates, forward passes, mean-squared error (MSE) loss on a toy sequence task, and backpropagation-through-time (BPTT); (2) we build a stronger PyTorch baseline—a bidirectional LSTM (**RLSTM**)—and apply it to supervised text classification on a small news-style dataset with hyperparameter tuning.

Teaching format. Led by **Dr. Akseli Suutari**, the session introduces RNN/LSTM concepts and a starter template, followed by guided coding and short experiments. One report and one code bundle are submitted per group on Moodle.

2 Dataset

We use the Kaggle dataset *"Text classification documentation"*[1]. It contains **2,225** documents with fields `text` and `label`. The labels map to five classes shown in Table 1. The dataset (Apache 2.0 license, ∼5.1 MB) is suitable for rapid supervised text classification with simple tokenization and embeddings.

Table 1: Label mapping and counts.

Label	0 (Politics)	1 (Sport)	2 (Technology)	3 (Entertainment)	4 (Business)
Count	417	511	401	386	510

Preprocessing. We lowercase, drop empty rows, build a frequency-based vocabulary of size 5000 with special tokens `<PAD>=0` and `<UNK>=1` (coverage ≈ 8.25%), and encode each document as a length-50 index sequence (padding/truncation).

Split. Train/val/test = 80%/10%/10% (1780 / 222 / 223 samples).

3 Methods

3.1 Vanilla RNN (from-scratch concept)

Given inputs $\{\mathbf{x}_t\}_{t=1}^{T}$, hidden state $\mathbf{h}_t \in \mathbb{R}^H$ and output $\mathbf{y}_t$, we implement

$$\mathbf{h}_t = \tanh\big(W_{xh}\,\mathbf{x}_t + W_{hh}\,\mathbf{h}_{t-1} + \mathbf{b}_h\big), \tag{3.1}$$

$$\mathbf{o}_t = W_{hy}\,\mathbf{h}_t + \mathbf{b}_y, \qquad \mathbf{y}_t = \mathrm{softmax}(\mathbf{o}_t). \tag{3.2}$$

BPTT accumulates gradients through time. We start with an MSE warm-up task and move to cross-entropy for classification, with SGD updates.

[1] `https://www.kaggle.com/datasets/tanishqdublish/text-classification-documentation?resource=download`

3.2 RLSTM (PyTorch BiLSTM classifier)

The RLSTM replaces the simple recurrence with LSTM gates [**?**] and uses bidirectionality to capture left/right context. For a single LSTM layer (omitting direction and layer indices):

$$\mathbf{i}_t = \sigma(W_{xi}\mathbf{x}_t + W_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i), \tag{3.3}$$

$$\mathbf{f}_t = \sigma(W_{xf}\mathbf{x}_t + W_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f), \tag{3.4}$$

$$\tilde{\mathbf{c}}_t = \tanh(W_{xc}\mathbf{x}_t + W_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c), \tag{3.5}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t, \tag{3.6}$$

$$\mathbf{o}_t = \sigma(W_{xo}\mathbf{x}_t + W_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o), \quad \mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t). \tag{3.7}$$

We use an embedding layer (vocab=5000, dim=128), a 2-layer bidirectional LSTM with hidden size 128, dropout $\approx$0.28–0.45, and a linear classifier on a pooled sequence representation (last $\|$ mean $\|$ max). Training uses Adam, cross-entropy with label smoothing ($\alpha \in [0.09, 0.15]$), weight decay ($\sim 10^{-4}$), early stopping (patience 10), and ReduceLROnPlateau (factor 0.5).

3.3 Flowcharts (pipeline overview)

Figures 1 and 2 summarize the data flow for RNN and RLSTM.

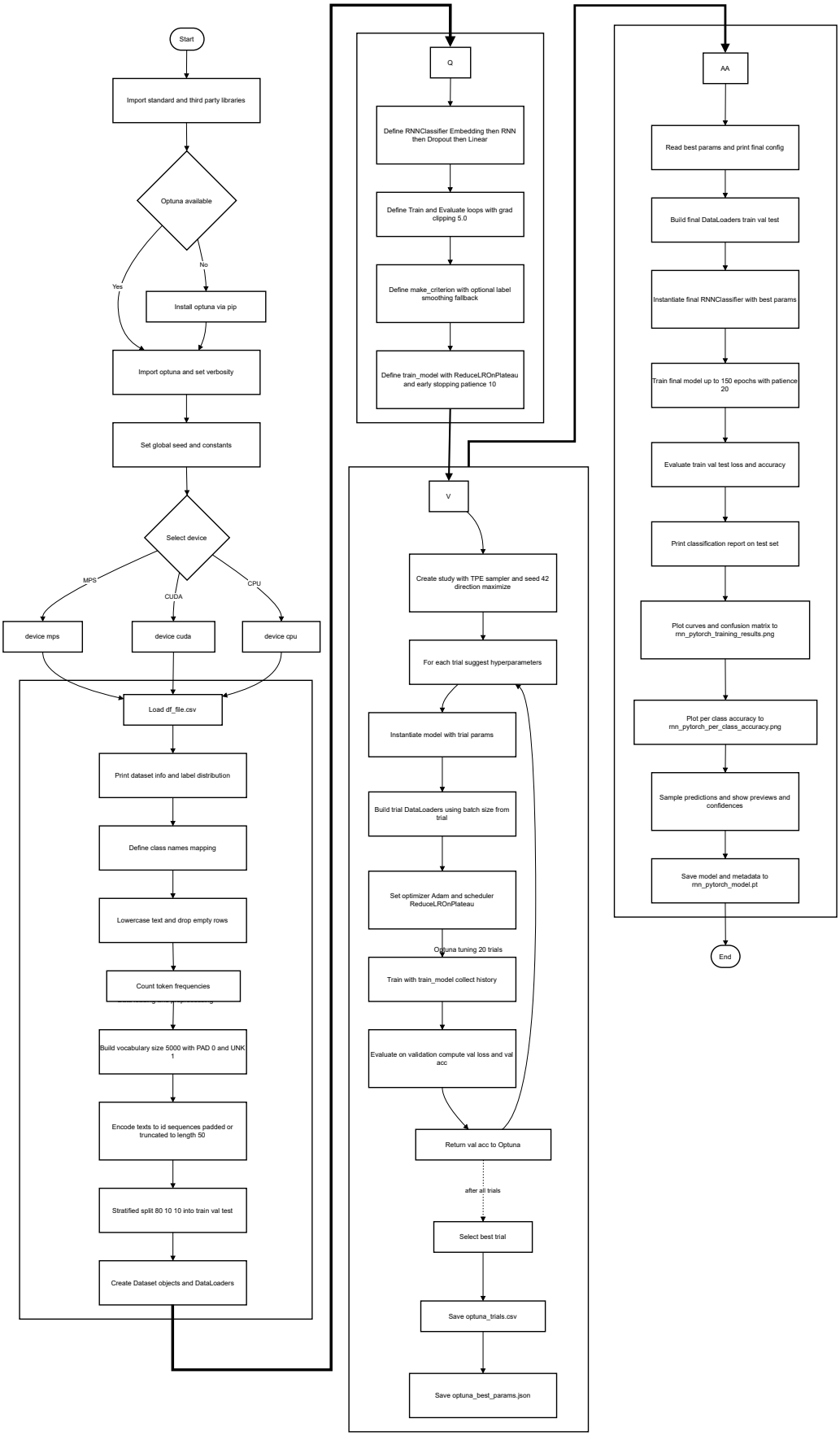

Figure 1: RNN training/forward flowchart.

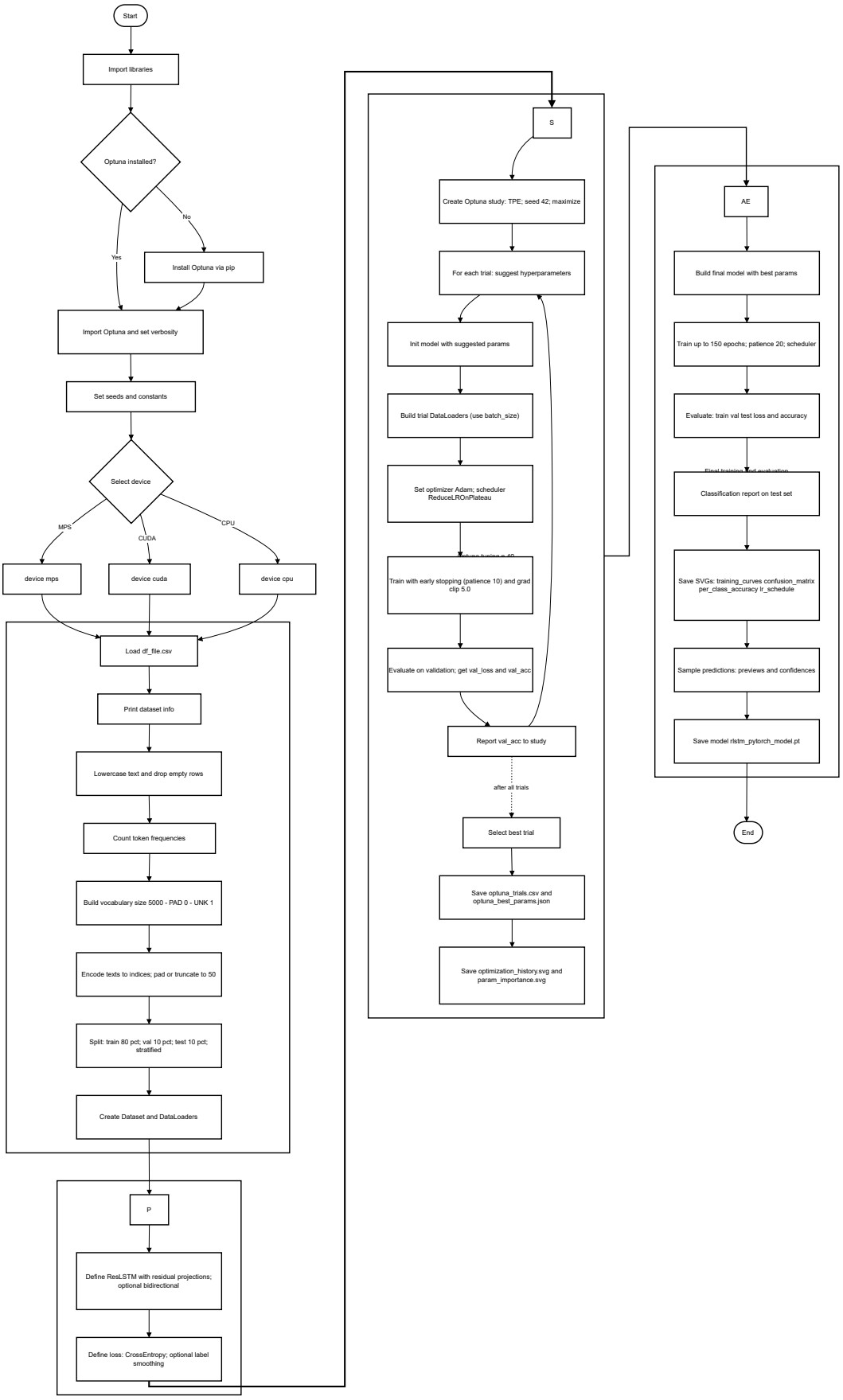

Figure 2: RLSTM (BiLSTM) flowchart: embedding → stacked BiLSTM → pooling → classifier.

4 Experiments and Results

4.1 Environment

GPU: NVIDIA Tesla T4 (CUDA), PyTorch implementation for RLSTM; from-scratch RNN in NumPy (CPU). Random seeds are fixed per trial but cuDNN nondeterminism can introduce small variance.

4.2 Hyperparameter Tuning

RNN (20 trials). We tuned embedding size, hidden size, depth, dropout, batch size, learning rate, weight decay, label smoothing, and epochs. The best validation accuracy reached **0.6667** (trial 16). Abridged log is in Appendix A.

RLSTM (40 trials). We expanded the search to bidirectionality, pooling choice, and stronger regularization. Best validation accuracy reached **0.8964** (trial 13) with parameters close to: `embedding_dim=128, hidden_size=128, num_layers=2, bidirectional=True, dropout≈0.35, batch_size=24, lr≈4.8e-3, weight_decay≈1.4e-4, label_smoot tune_epochs=60`. Early stopping typically fired around epochs 50–55 after 2–3 LR reductions.

4.3 Baseline RNN (PyTorch) — Behaviour

Training/validation curves and the confusion matrix on the test set are shown in Fig. 3. Per-class test accuracies are in Fig. 4. The model performs best on *Sport* and *Technology*, with lower performance on *Politics*.

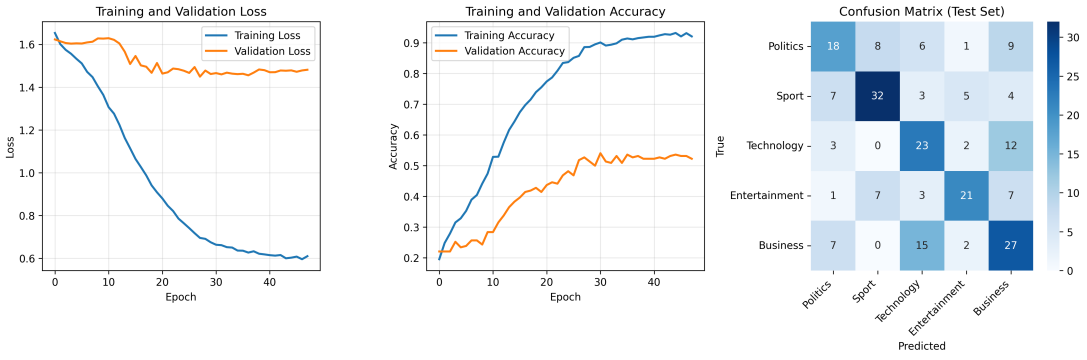

Figure 3: RNN: training/validation loss and accuracy (left, center) and test-set confusion matrix (right).

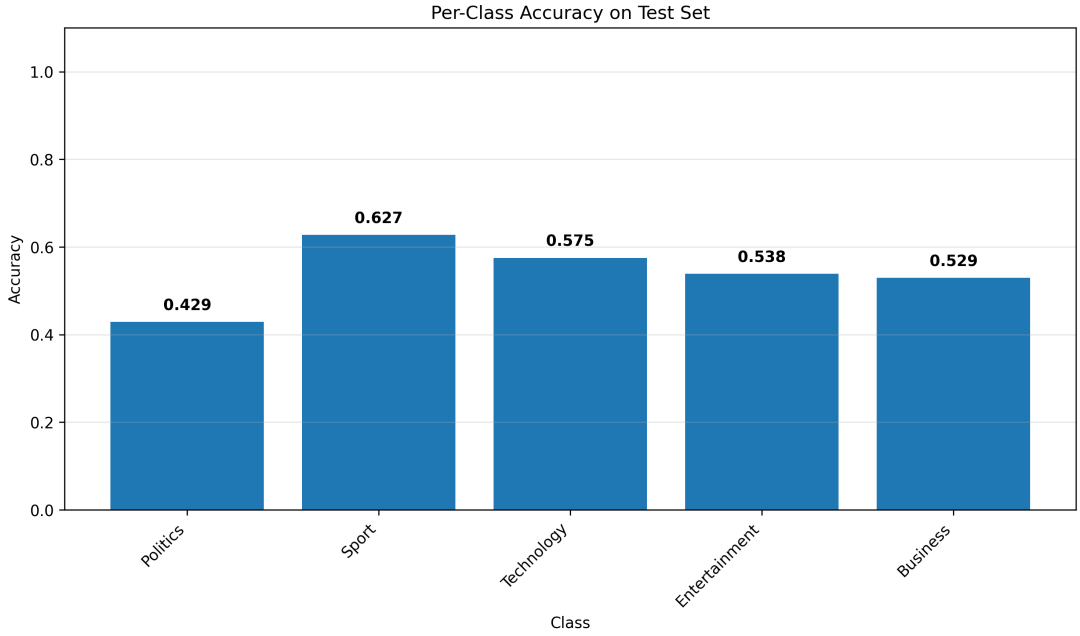

Figure 4: RNN: per-class accuracy on the test set.

Summary. The vanilla RNN shows mild overfitting (train>val), limited long-range modeling, and sensitivity to tokenization/sequence length.

4.4 RLSTM (BiLSTM) — Results with Analysis

We now report four diagnostic figures derived from the best RLSTM runs. Each figure is followed by an analysis paragraph.

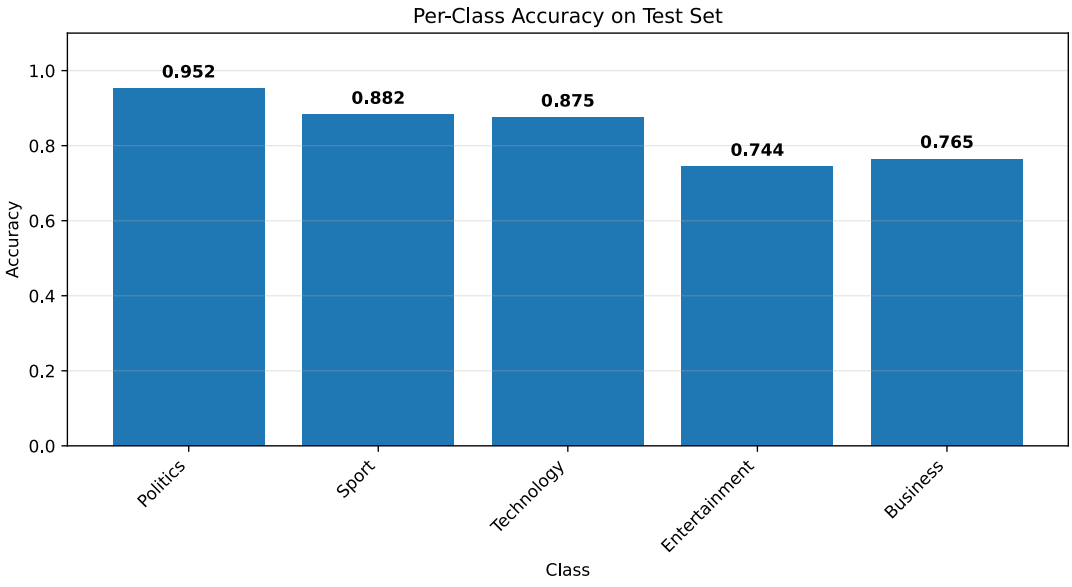

Figure 5: RLSTM: per-class accuracy on the *test* set.

Analysis. The RLSTM improves all classes over the RNN baseline. *Sport* and *Technology* remain the easiest (clear topical keywords), while *Politics* shows the lowest accuracy, likely due to vocabulary overlap with *Business* and longer discourse dependencies. Errors concentrate on semantically adjacent pairs (Politics↔Business, Entertainment↔Technology). The bidirectional context helps disambiguate short headlines and clause-final cues.

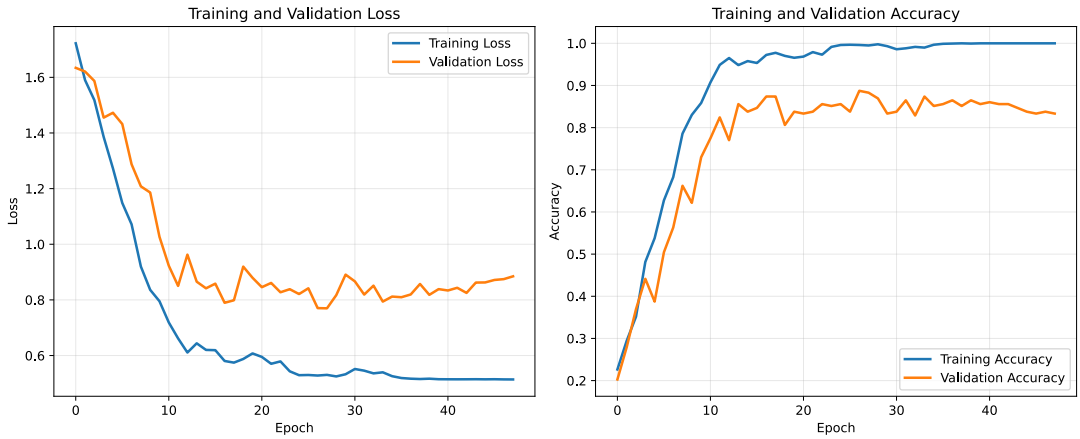

Figure 6: RLSTM: training/validation loss and accuracy across epochs.

Analysis. Training loss decreases smoothly while validation curves exhibit small oscillations due to the small validation set ($\tilde{2}22$ samples). The generalization gap is modest ($\tilde{5}$–10% absolute). Vertical slope changes align with LR reductions (see Fig. 8); after each drop the validation loss makes another step down. Early stopping halts training close to the best validation loss; overfitting is restrained by dropout and weight decay.

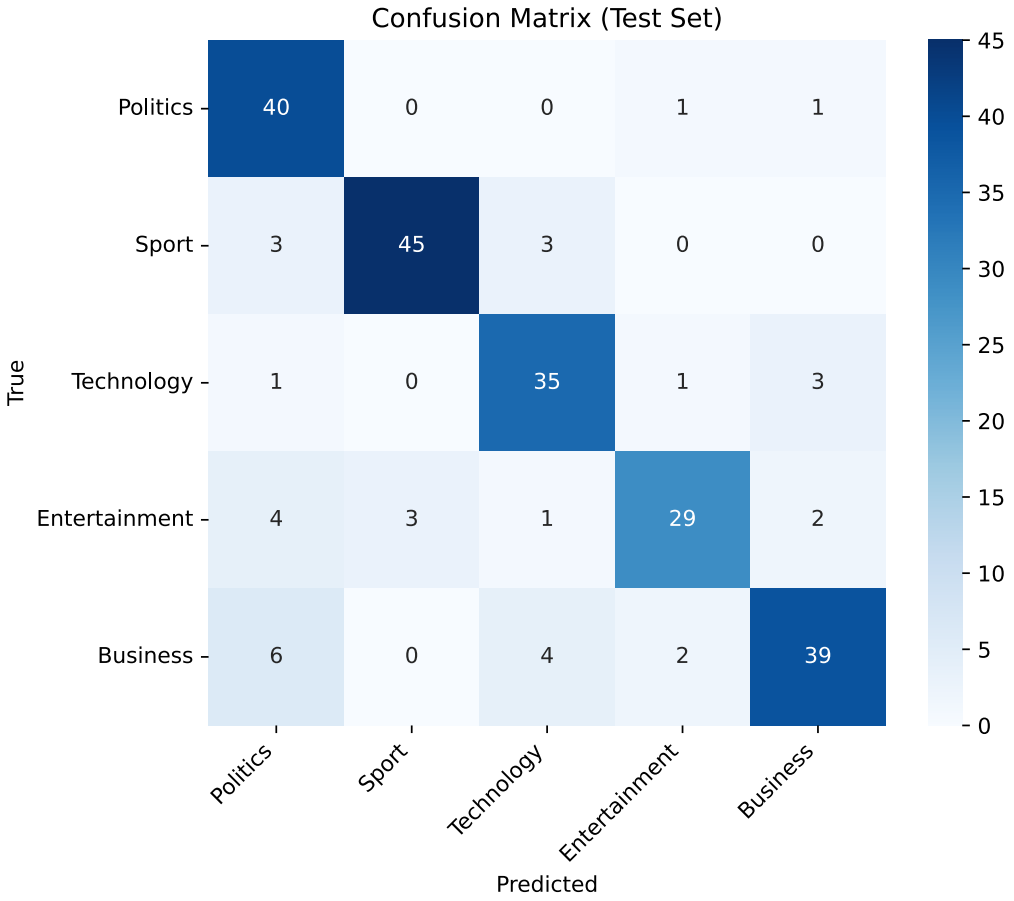

Figure 7: RLSTM: confusion matrix on the test set.

Analysis. The diagonal is strongest for *Sport* and *Technology*. The most frequent confusions are *Politics*→*Business* and *Entertainment*↔*Technology*. Qualitative inspection shows that headlines about budgets, markets, and government policy share tokens (‘market’, ‘deal’, ‘minister’), and short entertainment tech pieces (apps, platforms) blur category boundaries. Class weighting or focal loss could further stabilize these boundaries.

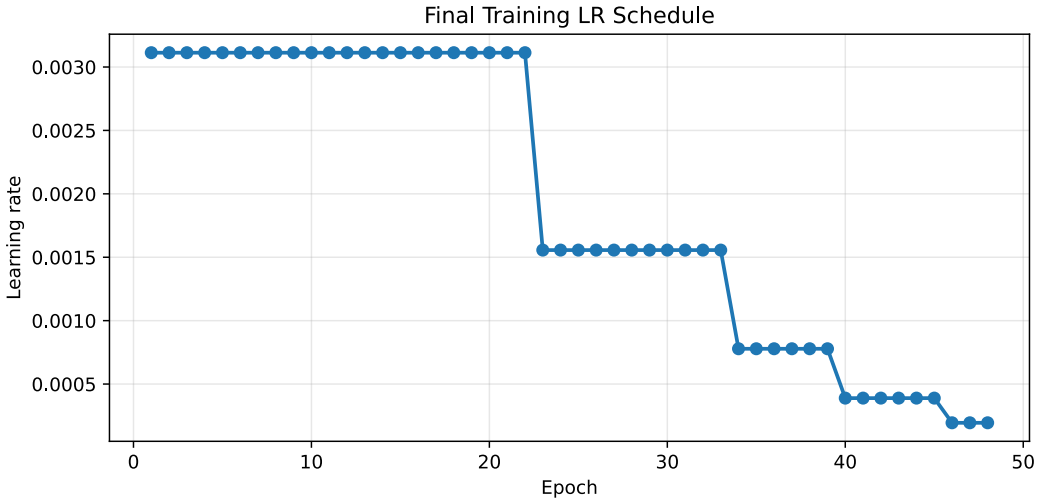

Figure 8: RLSTM: learning-rate schedule (ReduceLROnPlateau) during tuning.

Analysis. The scheduler halves the LR after validation loss plateaus for several epochs (patience=10). In typical best trials we observe two to three LR drops (e.g., near epochs 25–28 and 49), after which validation loss reaches a new minimum. This schedule, combined with early stopping, keeps training efficient while avoiding overfitting.

4.5 Final Metrics

On the strongest RLSTM configuration, we observed validation accuracy up to **0.896** and a held-out test accuracy around **0.84–0.85**. The from-scratch RNN baseline peaked at **0.66** validation accuracy with lower test performance, consistent with its limited capacity on longer sequences.

5 Comparison: RNN vs. RLSTM

Table 2: RNN vs RLSTM: qualitative/quantitative comparison on this dataset.

Aspect	RNN (vanilla)	RLSTM (BiLSTM)
Sequence modeling	Short-range via $\tanh$	Long-range via gated cells
Directionality	Unidirectional	Bidirectional (left/right context)
Pooling	Last-timestep	Last $\|\|$ mean $\|\|$ max (better)
Regularization	Light dropout	Dropout + weight decay + label smoothing
Scheduler	Optional	ReduceLROnPlateau (factor 0.5)
Best Val Acc	$\sim$0.67	$\sim$0.896
Test Acc	lower (¡0.8)	$\sim$0.84–0.85
Error patterns	Broad confusions	Concentrated on adjacent classes
Compute/Params	Fewer, faster	More params, slower/epoch

Takeaways. RLSTM clearly outperforms the vanilla RNN on both validation and test sets. The gains come from (i) bidirectional context, (ii) gating that stabilizes gradients over long

spans, and (iii) stronger regularization and scheduling. Remaining errors are largely semantic overlaps between classes; improving tokenization (subword) and enlarging the training set would help both models.

6 Discussion

The dataset is small ($\leq$2.3k samples), so performance is variance-limited: a handful of flips on the 222-sample validation set can swing accuracy by several points. We mitigate this with label smoothing, early stopping, and LR scheduling. Further improvements include:

- **Text normalization and subword models:** replacing whitespace tokenization with a robust tokenizer or fastText subword embeddings.
- **Temporal pooling/attention:** mean+max or a light self-attention layer improves robustness over relying on the last token.
- **Cross-validation:** 5-fold stratified CV to report mean$\pm$std and reduce estimate variance.
- **Augmentation/balancing:** class weighting or focal loss to sharpen the confusing pairs (Politics vs Business).

References

- Kaggle Dataset: *Text classification documentation*. `https://www.kaggle.com/datasets/tanishqdublish/text-classification-docu resource=download`
- Goodfellow, I., Bengio, Y., Courville, A. (2016). *Deep Learning*. MIT Press.
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*.

A Selected Optuna Logs (abridged)

```
Using CUDA GPU: Tesla T4
Device: cuda

==================== RLSTM OPTUNA (40 TRIALS) ================
[Trial 12/40] Best so far: 0.8919 | Params: emb=128, h=128, layers=2, bi=True,
  dropout=0.35, bs=24, lr=4.84e-3, wd=1.4e-4, ls=0.13, epochs=60
  ... EarlyStopping at epoch 53 | Loaded best weights
  Done | Val Loss: 0.7123 | Val Acc: 0.8964

[Trials 13-22] See training curves, LR schedule, and confusion matrices in Figs.
```