Assignment 4 – Text Data

Name: Akash Nair Eruvattu(811358511)

1. Problem Overview

Natural-language text is one of the richest and most unstructured forms of data available. Unlike images or tabular data, the meaning of a sentence depends not only on individual words but also on their sequential order and contextual relationships. Traditional bag-of-words or TF–IDF models fail to preserve these dependencies, making them insufficient for tasks such as sentiment analysis.

Recurrent Neural Networks (RNNs) address this issue by processing sequences word-by-word while maintaining a **memory of past information** through recurrent connections. This enables RNNs to learn dependencies such as negation ("not good") or emphasis ("absolutely fantastic") that static feature-based models cannot capture.

This assignment focuses on **sentiment classification** of IMDB movie reviews — determining whether a given review expresses a positive or negative opinion. Two distinct strategies for representing text are compared:

- **Custom-trained embeddings**, where word vectors are learned during training and adapt specifically to the IMDB domain.
- **Pretrained embeddings** (GloVe 6B.100d), which are general-purpose vectors trained on a massive external corpus to capture semantic similarity (e.g., "good" ≈ "great").
    - The investigation explores how these representations influence the model's ability to learn sentiment patterns, particularly under conditions of limited training data, and evaluates how performance scales as the data volume increases.

2. Objectives

The experiment pursues both **analytical** and **practical** goals:

- **Implement RNNs for text data:** Demonstrate how sequential deep-learning models can process text effectively.
- **Compare embedding strategies:** Quantitatively and qualitatively evaluate how a task-specific embedding differs from a general pretrained embedding in terms of performance, stability, and learning behavior.
- **Assess data sufficiency:** Determine how many training samples are required before a custom embedding begins outperforming a fixed pretrained one.
- **Identify overfitting characteristics:** Examine fluctuations in validation accuracy and loss to understand the impact of data scarcity and parameter count.
- **Derive generalization insights:** Formulate guidelines on when to use pretrained embeddings versus learning them from scratch.

This dual-focus approach combines theoretical understanding with empirical evidence to reach data-driven conclusions.

3. Dataset and Preprocessing

3.1 Dataset Description

The **IMDB movie review dataset**, available via keras.datasets.imdb, contains **50,000 reviews**, each labeled 1 = positive or 0 = negative.

The dataset is already tokenized, where every word is represented by an integer index based on overall frequency in the corpus. The most frequent word receives the lowest index.

3.2 Rationale for Preprocessing

Proper preprocessing is crucial for both computational efficiency and model performance. Limiting vocabulary size avoids excessive dimensionality and reduces the risk of overfitting to rare words. Padding ensures equal sequence length, enabling vectorized batch processing by GPUs.

3.3 Steps Performed

1. **Vocabulary Restriction:**
   Only the **top 10 000 most frequent words** were retained (num_words = 10000). Words outside this range were replaced by a reserved "OOV" (index 2). This balances coverage and model complexity.
2. **Sequence Truncation and Padding:**
   All reviews were standardized to **150 tokens** using pad_sequences(maxlen = 150). Longer reviews were truncated, while shorter ones were left-padded with zeros. This length was chosen to capture enough context without overwhelming the RNN's memory.
3. **Dataset Splitting:**
   - Training sets – varied sizes (100, 3 000, 7 000, 10 000)
   - Validation set – 10 000 samples
   - Test set – 15 000 samples (from remaining data)
     This setup allows fair performance comparison across different training volumes.
4. **Label Preparation:**
   Target values were binary-encoded as 0 or 1.
5. **Normalization for GloVe:**
   - Loaded glove.6B.100d.txt, containing 400 000 words × 100 dimensions.
   - Built an **embedding dictionary** mapping each word to its 100-D vector.
   - Constructed an **embedding matrix** (10 000 × 100) aligned with the IMDB indices.
   - Words not present in GloVe were assigned small random vectors from a uniform distribution (−0.05 to 0.05).

3.4 Rationale for Pretrained Embeddings

GloVe embeddings encode global co-occurrence statistics of words across billions of tokens, positioning semantically related words nearby in vector space. Theoretically, this provides a strong initialization, especially when labeled data is scarce.
However, since these vectors are trained on **news and Wikipedia corpora**, their effectiveness on **informal movie-review text** is uncertain — a key factor this assignment examines.

---

4. Model Architecture and Design Choices

4.1 Why an RNN and Why Bidirectional LSTM

A simple RNN captures sequential dependencies but suffers from vanishing gradients when handling long contexts. The **Long Short-Term Memory (LSTM)** architecture overcomes this by introducing gates (input, forget, and output) that regulate information flow.
A **Bidirectional LSTM** further enhances context comprehension by reading the sequence in both directions — essential for sentiment phrases like "not only good but brilliant" where key modifiers appear before and after target words.

4.2 Model Structures

Layer	Custom Embedding Model	Pretrained GloVe Model
Input	Review sequence (150 tokens)	Same

Layer	Custom Embedding Model	Pretrained GloVe Model
Embedding	Embedding(10000, 32, trainable=True) – randomly initialized	Embedding(10000, 100, weights=[embedding_matrix], trainable=False)
Recurrent	Bidirectional(LSTM(32))	Bidirectional(LSTM(32))
Output	Dense(1, activation='sigmoid')	Dense(1, activation='sigmoid')

4.3 Training Configuration

- **Optimizer:** Adam (learning rate = 0.001)
- **Loss Function:** Binary Crossentropy
- **Batch Size:** 32
- **Epochs:** 10 (with early stopping on validation loss)
- **Metrics:** Accuracy and Loss

4.4 Design Rationale

- A small embedding dimension (32) for custom embeddings prevents over-parameterization.
- The GloVe layer is fixed to isolate the effect of pretrained semantic knowledge.
- Bidirectional LSTM with 32 units balances capacity and generalization.

---

5. Experimental Results and Empirical Findings

Embedding Type	Training Size	Training Accuracy	Test Accuracy	Test Loss
Custom	100	0.98	0.49	0.69
	3 000	0.99	0.73	0.54
	7 000	0.97	0.84	0.34
	10 000	0.97	0.85	0.33
GloVe (Pretrained)	100	1.00	0.49	1.60
	3 000	0.97	0.49	1.94
	7 000	0.92	0.50	1.72
	10 000	0.88	0.49	1.34

5.1 Accuracy and Loss Trends

- **Custom Embeddings:** Show continuous improvement in validation and test performance as training data increases.

- **Pretrained Embeddings:** Maintain almost flat test accuracy near 49 %, indicating random guessing behavior.
- **Test Loss:** Declines for custom embeddings but remains high and erratic for GloVe.

5.2 Interpretation

- Custom embeddings learn sentiment-specific associations (e.g., “awful” → negative, “must-watch” → positive).
- Fixed GloVe vectors cannot adapt to slang, sarcasm, or domain-specific phrases (“Oscar-worthy,” “plot dragged”) not present in Wikipedia.

---

6. Visual Analysis and Behavioral Patterns

6.1 Training vs Validation Accuracy (Customed Embeddings)

- With small datasets (100–3 000 samples), the training curve reaches ~1.0 quickly, while validation accuracy stagnates around 0.5–0.7, showing overfitting.
  When increased to 7 000 and 10 000 samples, validation accuracy climbs and nearly aligns with training accuracy (~0.85), demonstrating that additional data reduces overfitting and enables the network to generalize to unseen patterns.

6.2 Training vs Validation Accuracy (GloVe)

- Across all training sizes, validation accuracy stays around 0.49–0.50 even as training accuracy peaks at 1.0. This imbalance implies that the model memorizes the limited examples but cannot leverage the static embeddings to classify new reviews correctly.

6.3 Comparative Bar Charts

- Bar plots reveal that as data size grows, the gap between the two approaches widens dramatically — proving that pretrained embeddings without fine-tuning can stagnate regardless of data availability when domain misalignment exists.

6.4 Loss Curves

- Custom models exhibit steadily declining loss with increasing samples, signifying improving prediction confidence.
  GloVe models show oscillating high loss, confirming that they are stuck in suboptimal local minima because the embedding weights are frozen and cannot adapt.

---

7. Detailed Discussion and Insights

7.1 Overfitting in Low-Data Scenarios

When only 100 training samples were used, both models displayed severe overfitting: the network memorized training examples but failed on validation data. This is expected, as RNNs contain tens of thousands of parameters yet were fed minimal data.
Regularization techniques (dropout in LSTM, L2 penalty) could mitigate this, but they were deliberately omitted to isolate the effect of embeddings.

7.2 Effect of Training Data Volume

Performance of the custom model scales almost logarithmically with sample size — rapid gains from 100 → 3 000 → 7 000 samples, then gradual saturation around 10 000. This reflects a typical learning-curve pattern where additional data yields diminishing but still positive returns.

7.3 Limitations of Pretrained GloVe Vectors

1. **Domain Mismatch:** GloVe trained on Wikipedia and Gigaword lacks sentiment-heavy movie-review terms like “plot-twist”, “underrated”, or “over-hyped”.
2. **Frozen Weights:** Setting trainable=False locks embeddings, preventing gradient updates that could align them to IMDB context.
3. **Semantic Ambiguity:** Words like “wicked” or “sick” can shift sentiment in colloquial contexts — pretrained vectors encode generic meanings that confuse classification.
4. **Dimensional Redundancy:** High-dimensional (100-D) vectors may capture semantic relations irrelevant to sentiment polarity, reducing discriminative power.

7.4 Learning Dynamics of Custom Embeddings

The custom embeddings gradually cluster words with similar sentiment into coherent vector neighborhoods (e.g., “excellent,” “superb,” “brilliant” near each other). This emergent structure enhances generalization and allows the LSTM to capture nuanced dependencies like “not so great” vs “not bad.”

7.5 Interpretability Considerations

A saliency-map inspection (optional step) would likely show that the custom model attends more strongly to emotionally charged words, while the GloVe model might distribute attention uniformly — another indicator of its lack of task specialization.

---

8. Conclusion

1. **Best Performing Model:** Custom-trained Embedding + Bidirectional LSTM
2. **Maximum Test Accuracy:** ≈ 85 % with 10 000 training samples
3. **Key Findings:**
   - Custom embeddings excel because they learn sentiment patterns tailored to the IMDB domain.
   - Pretrained GloVe embeddings, though linguistically rich, fail to adapt when frozen.
   - Model generalization improves steadily with additional training data.
   - The crossover point — where custom embeddings start outperforming — occurs as early as ~3 000 samples.
4. **Practical Recommendations:**
   - For domain-specific sentiment tasks, **train embeddings from scratch** or **fine-tune** pretrained ones.
   - Reserve frozen pretrained embeddings for extremely low-data scenarios with strong domain overlap.
   - Regularization and hyperparameter tuning can further stabilize learning as dataset size increases.
5. **Broader Implications:**
   The study demonstrates that **transfer learning is not universally beneficial**. Its success depends heavily on domain similarity, dataset size, and whether the pretrained parameters are allowed to adapt.

---

9. Learning Outcomes and Reflections

Through this assignment, several key competencies were developed:

1. **Conceptual Understanding:**
   - Grasped how RNNs process sequential data and why LSTM’s gating mechanisms mitigate gradient-vanishing issues.
   - Understood the role of word embeddings in transforming sparse token indices into dense semantic vectors.
2. **Technical Implementation Skills:**

- Practiced building embedding layers (trainable vs. frozen).
- Learned how to import and align external embeddings such as GloVe with Keras token indices.
- Gained proficiency in handling padding, truncation, and batching for sequence data.

3. **Analytical Evaluation:**
   - Interpreted learning curves and understood overfitting symptoms.
   - Quantified performance sensitivity to data volume and embedding initialization.
4. **Practical Reflection:**
   - Recognized that data quantity and domain relevance often outweigh the theoretical advantage of pretraining.
   - Developed an appreciation for experiment reproducibility and systematic documentation.
5. **Future Extensions:**
   - Fine-tune pretrained embeddings (trainable=True) to bridge the observed gap.
   - Experiment with **GRU**, **1D CNN**, or **transformer-based models** for better efficiency.
   - Integrate **attention mechanisms** to highlight sentiment-bearing words.
   - Use **transfer learning with domain-adaptive pretraining** (e.g., fine-tune BERT on IMDB).

---

10. Final Summary

Custom embeddings trained alongside the model consistently outperform static pretrained embeddings for IMDB sentiment analysis. Although pretrained vectors encode broad semantics, they require adaptation to be useful for specialized domains. As the quantity of domain-specific training data increases, task-specific learning surpasses general semantic knowledge, leading to superior accuracy, lower loss, and robust generalization.