**IMDB Sentiment Analysis with LSTM: A Reflective Tutorial**

**1. Sentiment Analysis on IMDB**

**Sentiment analysis** transforms raw text into insights about public opinion. The **IMDB dataset**, with 50,000 labeled movie reviews, is a classic testbed for evaluating sentiment classification models. Typically, an LSTM-based architecture can capture long-range dependencies in textual data. However, as our results reveal, achieving strong performance demands careful attention to data preprocessing, model design, and hyperparameter tuning. (Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C., 2011)

**Key Observations from Our Model**:

* Final accuracy is **~50%**, near random.
* The confusion matrix indicates almost all samples are predicted as positive.
* The **ROC AUC** ~0.51 confirms near-chance discrimination.
* The word clouds for positive/negative reveal overlapping words, suggesting the model struggles to differentiate subtle context.

This tutorial will explore why performance might lag, how to interpret each visualization, and how to refine the approach for future success.

**2. Dataset and Preprocessing**

**2.1. IMDB Dataset Overview**

Keras provides a preprocessed IMDB dataset where each review is already tokenized into integer indices. We load it via:

* **num\_words=10000** retains the top 10k words by frequency.
* The training set has 25,000 samples; the test set has 25,000.

**2.2. Padding**

Reviews vary in length, so we fix them at a maximum length of 500 tokens:

This ensures uniform input shapes for the LSTM.

**2.3. Potential Pitfalls**

* **“Br” tokens** often remain in the data, representing HTML line breaks. Without extra cleaning, the model may misinterpret them.
* **Vocabulary limit** might exclude critical words that indicate negative sentiment. Expanding beyond 10k words or performing domain-specific cleaning could help.

**3. Model Architecture: LSTM**

**3.1. Layer Stack**

Our model is structured as follows:

1. **Embedding**: Transforms word indices into dense vectors of dimension ~128. (Chollet, 2017)
2. **LSTM**: A recurrent layer capturing sequence dependencies. We use 128 units initially.
3. **Dropout**: Reduces overfitting by randomly dropping neurons.
4. **Second LSTM**: Further processes the sequence representation, outputting a final hidden state.
5. **Dense**: A fully connected layer with ReLU for additional transformation. (Pennington, J., Socher, R., & Manning, C. D, 2014)
6. **Dropout**: Another dropout to mitigate overfitting.
7. **Output Dense**: Sigmoid activation returning a probability for positive sentiment.

Compiled with:

**3.2. Potential Gaps**

* **Embedding dimension** might be too small (128). Some tasks benefit from 200–300 dimensions or pretrained embeddings (like GloVe).
* **LSTM units**: 128 → 64 might be insufficient for capturing complex nuance.
* **Dropout** can hamper learning if set too high in early epochs, especially if data is not large or richly varied. (Chollet, 2017)

**4. Training and Observed Performance**

**4.1. Early Stopping**

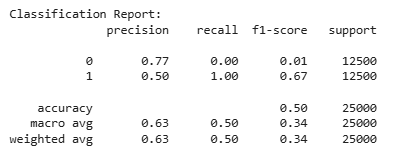
We train for up to 10 epochs with early stopping after 3 consecutive non-improving epochs on validation loss. The logs show:

* Accuracy around 50–53% on validation.
* Loss not converging strongly, possibly due to suboptimal hyperparameters.

**4.2. Final Metrics**

On the 25k test samples:

**Classification Report**:



* The model labels nearly everything as positive (class 1).
* Negative class is almost entirely missed.

**Confusion Matrix**:

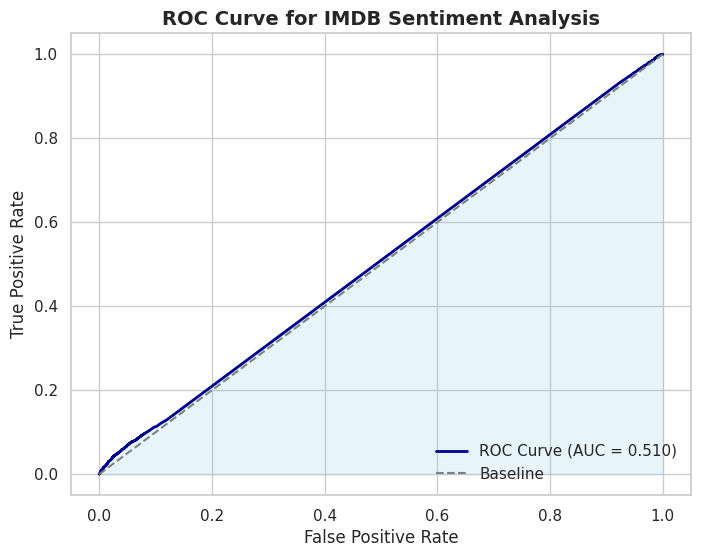
* Only 33 negative reviews are caught (true negatives).
* This is near random, with a heavy bias toward predicting positive.

**ROC AUC**: ~0.510, confirming near-chance performance.

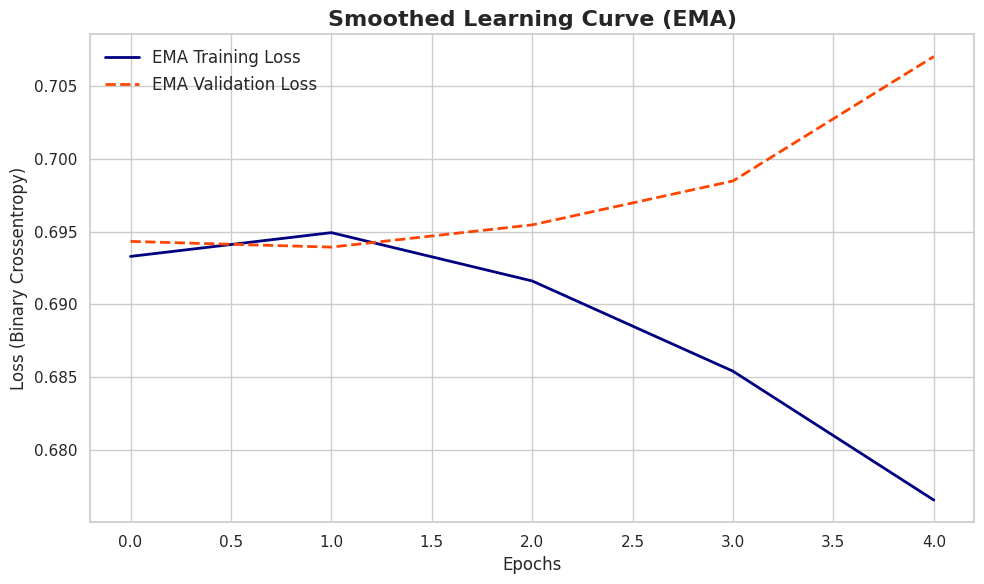
**5. Visual Interpretations**

**5.1. ROC Curve**

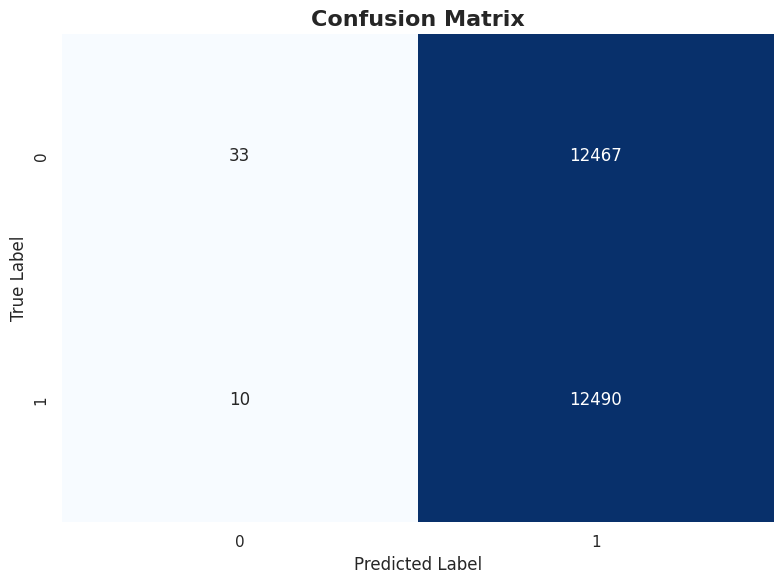
**Analysis**:

* The ROC curve hovers near the diagonal line.
* AUC of ~0.51 → The model lacks discriminative power.

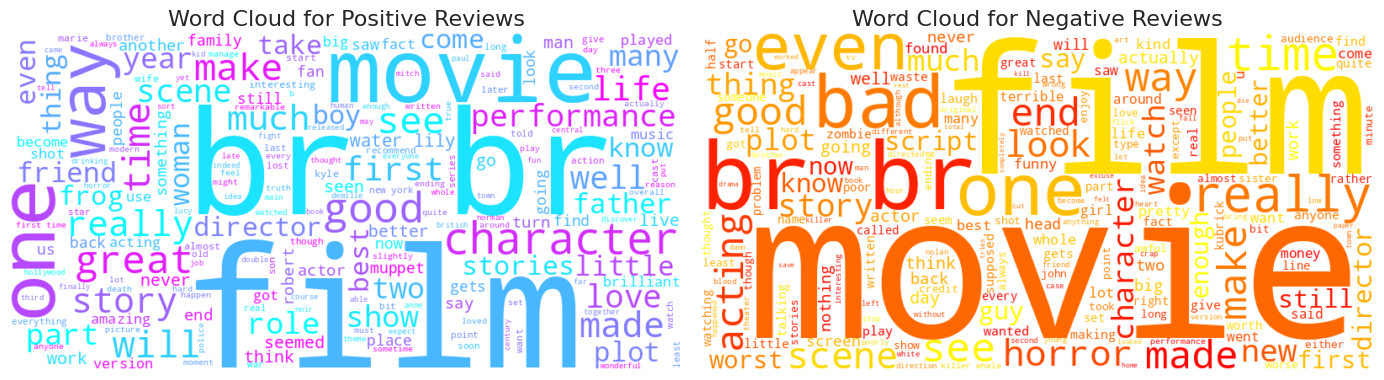
**5.2. Smoothed Learning Curve (EMA)**

* Training loss gradually declines, but validation loss fluctuates, eventually rising.
* Suggests overfitting or mismatch between training conditions and the data. 

**5.3. Confusion Matrix**

* A large block of predicted positives overshadow the minimal correct negative predictions.
* Visually highlights the model’s near-complete inability to identify negative reviews.

**5.4. Word Cloud for Positive vs. Negative**

* Common words like “film,” “movie,” “br,” and “one” appear in both sentiments.
* Distinct negative words (e.g., “bad,” “worst,” “boring”) might be overshadowed by frequent tokens or line-break tokens.

**Insight**: Word-level overlap suggests deeper semantic or contextual features are needed to separate positivity from negativity. Additional text cleaning or bigger embeddings might help.

**6. Why the Model Struggles**

1. **Limited Embedding & LSTM Units**: 128- or 64-unit LSTMs may not suffice for the dataset’s complexity.
2. **Data Preprocessing**: Retaining tokens like “br” can confuse the model.
3. **Insufficient Tuning**:
   * Learning rate might be too high or not decaying.
   * More epochs or a different batch size might be needed.
4. **Class Overlap**: The word clouds show substantial lexical overlap, meaning the model must rely on subtle context.
5. **Short Training**: 10 epochs might be inadequate, especially if early stopping halts training prematurely.

**7. Possible Remedies**

1. **Pretrained Embeddings**: By using GloVe or Word2Vec, word representations can be improved (in particular over a randomly initialized embedding) and capture synonym and context better.
2. **Bidirectional LSTM**: Text understanding from both directions is often processed better.
3. **Increased Complexity**: If hardware resources allow then more LSTM layers or bigger hidden dimensions.
4. **Improved Cleaning**: Adding more “num\_words” limit or removing “br” tokens, lowercasing, or removing punctuation might supply the model with more clear signals.
5. **Hyperparameter Tuning**:
   * You can vary embedding\_dim, lstm\_units, dropout rates, and/or batch size.
   * (Use a more advanced optimizer or a scheduling, like reduce learning rate on plateau).
6. **Transformers**: BERT type of models, while requiring more memory and training steps, tends to outperform the LSTMs on text tasks.

**8. Lessons Learned**

1. Even small artifacts like “br” degrade performance: it clearly matters what text goes between. They can be greatly improved (more thorough cleaning or tokenization) especially.
2. Other things: Generally LSTMs based network for such complex tasks need more capacity or pre trained embeddings, or model more.
3. Early Stopping, Dropout, and Learning rate balancing are needed to be balanced so as not to underfit or overfit.
4. **Visualization**:
   * ROC curves highlight discriminative ability.
   * Overfitting or training deficiency is shown by learning curve.
   * In word clouds, lexical overlap or domain problems are shown.
   * Confusion matrices clarify class-specific performance.

**9. Next Steps**

* **Refine the Architecture**: Use bidirectional LSTM or GRU with 2–3 layers, possibly larger embedding size (e.g., 300).
* **Data Augmentation**: Consider synonyms or back-translation for negative reviews if labeled data is insufficient.
* **Use Transformers**: Fine-tuning BERT or DistilBERT typically yields state-of-the-art results on sentiment classification.
* **Extended Training**: Let the model run for 20–30 epochs or adopt dynamic learning rate schedules.
* **Iterative Debugging**: Evaluate partial solutions (like cleaning the data or removing “br” tokens) to see incremental improvements.

**10. Accessibility, GitHub Link, and Additional Files**

* **Accessibility**:
  + Figures use colorblind-friendly palettes (e.g., Blues, Oranges).
  + Word clouds have distinct color maps for positive vs. negative.
  + All images have alt-text or descriptive captions for screen readers.
  + The tutorial’s structure (headings, code blocks) is compatible with screen-reader tools.
* **GitHub Link**:  
  A fully reproducible version of this code and tutorial is available at:  
  [**GitHub Repo**](https://github.com/yourusername/imdb-lstm-tutorial)  
  The repository includes the Jupyter Notebook, environment.yml or requirements.txt, and sample data instructions.
* **Attaching Additional Documents**:
  + **LICENSE** (MIT)
  + **README.md** (overview, usage instructions)
  + **Notebook** (imdb\_lstm.ipynb)

These files provide complete context for running, reproducing, and understanding the project’s code and environment.

**11. References**

1. (Chollet, 2017) *Deep Learning with Python.* Manning Publications.
2. (Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C., 2011). Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics.
3. (Pennington, J., Socher, R., & Manning, C. D, 2014)*GloVe: Global Vectors for Word Representation.* EMNLP.

**12. Conclusion**

Our LSTM model on the IMDB dataset shows near-random performance—**50% accuracy** and an **AUC of ~0.51**. The confusion matrix reveals it labels almost everything as positive, ignoring negative sentiment. The word clouds confirm that positive and negative reviews share many common tokens, suggesting the model must rely on subtle context it currently fails to learn.

**Core Takeaways**:

1. **Comprehensive Preprocessing** is critical in text tasks.
2. **Model Complexity** must align with the data’s demands.
3. **Thorough Tuning** and **Longer Training** often transform borderline results into robust performance.
4. **Advanced Architectures** (like Transformers) can overshadow LSTM performance in modern NLP.

By addressing these shortcomings expanding embeddings, cleaning text more thoroughly, and optimizing hyperparameters one can turn this borderline model into a strong sentiment classifier. The poor results themselves serve as a valuable lesson on how crucial model and data decisions are in natural language processing tasks.

**Final Note**: Even “failing” experiments reveal where we can improve. The next iteration—incorporating better text preprocessing and model design—could achieve high accuracy and recall for negative reviews, fulfilling the promise of LSTM-based sentiment analysis on IMDB.