CSE587 Spring 2025 Midterm Project
Sentiment Analysis with RNNs and CNNs

Jiamu Bai **Ryo Kamoi**

GitHub Repository: `https://github.com/jiamubai/cse587-midterm`

1 Introduction

In this project, we investigate the performances of RNNs [1, 3] and CNNs [4] on sentiment analysis [7, 6]. Prior study [10] suggests that CNNs are often competitive with or better than RNNs on sentiment analysis. We further explore a detailed comparison between their performances in multiple aspects. Specifically, we evaluate CNNs, naive RNNs, and LSTMs to examine (1) performance differences among different architectures, (2) hyperparameters that influence their performance on sentiment analysis, and (3) factors in input data that make sentiment analysis difficult for RNNs and CNNs.

2 Problem Definition and Dataset Curation

This section provides the problem definition of sentiment analysis and details of our target datasets.

2.1 Problem Definition: Sentiment Analysis

Sentiment analysis (or sentiment classification, sentiment categorization) is a classification task to label texts with their sentiment [7, 6]. The target labels can be binary (i.e., positive vs. negative) [7, 9, 5], three options (i.e., positive, negative, or neutral) [2], or integer scores (e.g., 0 to 5) [6, 11].

2.2 Dataset

Dataset Curation. In this project, we use three publicly available sentiment analysis datasets, which are listed below. We collected the datasets from the following URLs.

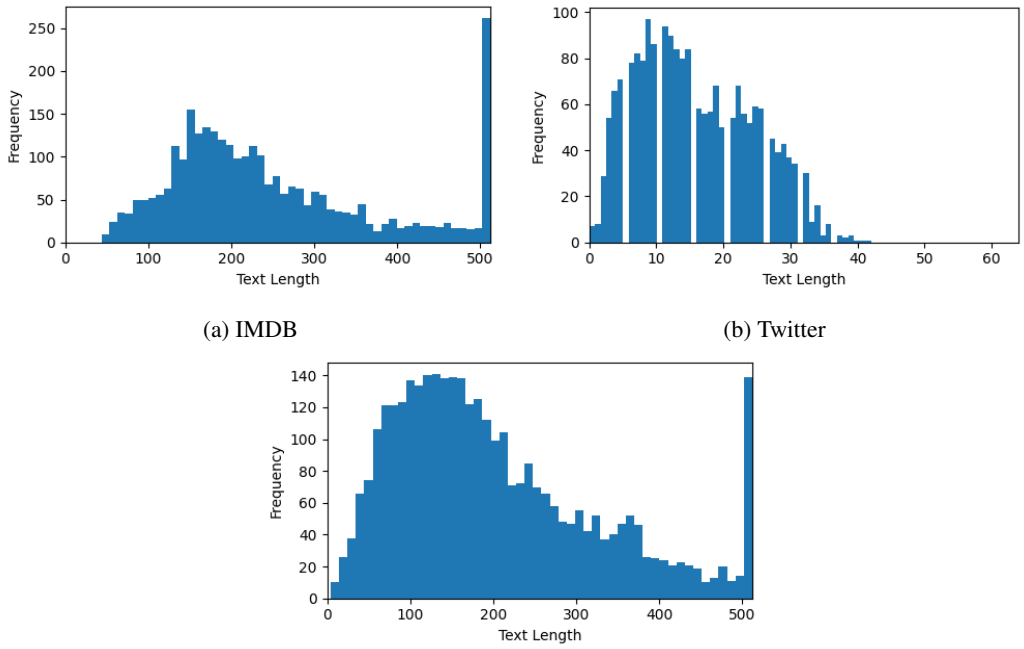
- **IMDB Reviews** [5][1] consists of informal movie reviews from the Internet Movie Database (IMDB).
- **Twitter Sentiment140** [2][2] consists of Twitter messages with emoticons, which are used as noisy labels for sentiment classification. We sample 10K messages from the original dataset.
- **Yelp Reviews** [11][3] consists of reviews from Yelp, which publishes crowd-sourced reviews about restaurants and businesses. It is extracted from the Yelp Dataset Challenge 2015 data.

Dataset Statistics. Table 1 provides dataset statistics. Each dataset we evaluate has around 10k to 15k instances. The IMDB and Twitter datasets include binary classification tasks, and the Yelp dataset consists of a multiple-choice classification task (1 to 5). Texts in the Twitter and Yelp datasets include around 200-250 tokens on average, while texts in the IMDB dataset are much shorter (16.4 tokens in average).

[1] `https://huggingface.co/datasets/stanfordnlp/imdb`
[2] `https://www.kaggle.com/datasets/kazanova/sentiment140`
[3] `https://huggingface.co/datasets/Yelp/yelp_review_full`

Dataset	Labels	# Instances	Ave. # Tokens
IMDB	Binary	15K	16.4
Twitter	Binary	10k	267.0
Yelp	1 to 5	17,336	208.2

Table 1: Dataset Statistics



(a) IMDB

(b) Twitter

(c) Yelp

Figure 1: Number of tokens of sentiment analysis dataset. These figures show histograms after truncation at 512 tokens.

Dataset Preprocessing We tokenize the input texts using `word_tokenize` in NLTK. We set the maximum length to 512 tokens and added paddings (word vector with zero elements) with the left padding (padding tokens before the text). Figure 1 shows details of their text lengths.

3 Experimental Settings

This section explains word embeddings, algorithms, and the training strategies we used.

3.1 Word Embeddings

We compare two sizes of GloVe: GloVe-50 and GloVe-300 [8], where each word is mapped to a unique vector with length 50 and 300, respectively.

3.2 Models

We evaluate CNNs [4], Vanilla RNNs [1], and LSTMs [3]. For CNNs, we use a single convolution layer followed by ReLU, max pooling, and a fully connected layer. For RNNs and LSTMs, we use a single layer and use the final hidden state for classification.

We evaluate models with the following parameters.

- Hidden state size: 128, 256, 512

- (only for CNNs) Kernel size: 3, 5, 7

3.3 Training

We use PyTorch in our experiments. We train all models with mini-batch gradient descent in the following settings.

- Learning rate: 1e-3
- Optimizer: Adam
- Number of Epochs: 20
- Batch Size: 1024

4 Results and Analysis

This section provides our results and analysis. Table 2 shows our main result.

Dataset	CNN	RNN	LSTM
IMDB	65.7	57.3	66.0
Twitter	67.7	59.0	69.5
Yelp	46.9	46.4	45.8

(a) GloVe-50

Dataset	CNN	RNN	LSTM
IMDB	74.0	50.2	72.1
Twitter	69.8	50.7	71.2
Yelp	49.2	46.1	42.3

(b) GloVe-300

Table 2: Accuracy of sentiment analysis under different datasets and models. The hidden state size is 512, and the kernel size for CNN is 7. The results are from left padding.

Influence of architectures. Table 2 shows that CNNs are competitive with LSTMs, which is consistent with the observation in prior work [10]. We observe that LSTMs are slightly better on Twitter, but CNNs are slightly better on Yelp. We consider this due to the property of the two datasets, where Yelp includes less diverse expressions than Twitter, as CNNs may be better at simple phrase detection. We observe that naive RNNs are consistently worse than both models.

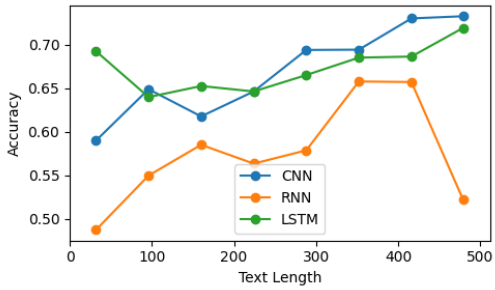
Influence of word embeddings. Table 2 shows that GloVe-300 improves performance from GloVe-50 in most cases of CNNs and LSTMs, but it harms the performance of RNNs. This result suggests that larger word embeddings provide more information but can make the training of RNNs difficult, and more careful hyperparameter tuning might be required to achieve a better goal.

Influence of text length. Figure 1 shows relationships between text length (number of tokens) and performance on semantic analysis in accuracy. We observe that different datasets exhibit different trends in the relationship between text length and sentiment analysis performance. Specifically, longer texts are easier in IMDB, but there is no clear relationship in other datasets.

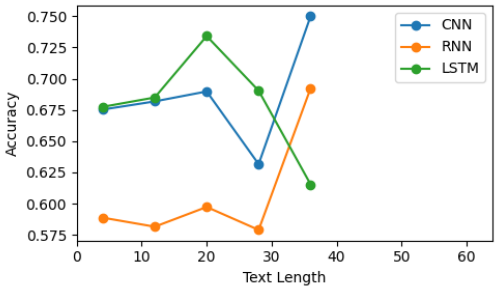
We expected that longer text is difficult, especially for naive RNNs, but our observations oppose our hypothesis. We consider that this result is due to the property of sentiment analysis in these datasets, which often does not require an understanding of the whole text.

Influence of hidden states size. Figure 3 shows the relationships between hidden state sizes and accuracy on sentiment analysis. Our results show that larger hidden states do not consistently improve performance and even harm the performance in some cases, especially for RNNs (on IMDB and Twitter). Our observation suggests that the hidden state size is not a bottleneck in this task and can make training difficult or lead to overfitting.

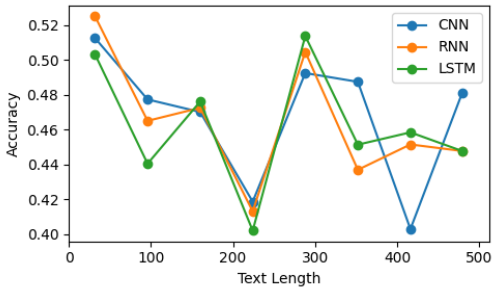
Influence of kernel size of CNNs. Figure 4 shows that the kernel size of CNNs does not have a large influence and slightly harms the performance in accuracy. This result is consistent with our observation in text length, suggesting that sentiment analysis in these datasets does not require an understanding of the whole text, and detecting short phrases is often sufficient.



(a) IMDB

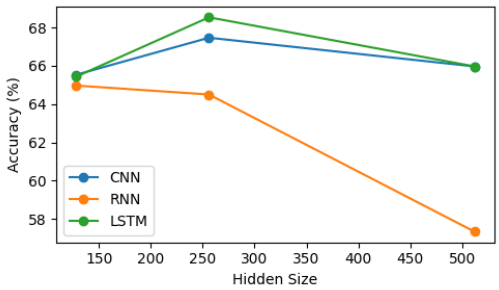


(b) Twitter

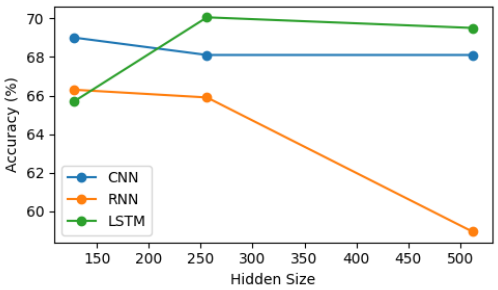


(c) Yelp

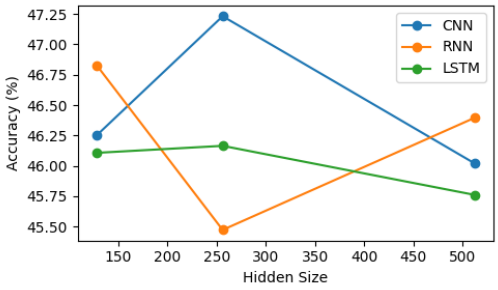
Figure 2: Accuracy vs. text length (word embedding size = 50, hidden state size = 512, CNN kernel size = 7).



(a) IMDB



(b) Twitter



(c) Yelp

Figure 3: Accuracy vs. hidden state size (word embedding size = 50, CNN kernel size = 7).

Padding strategies. We observe that left padding (adding padding tokens before input text) works much better than right padding (adding padding tokens after input text) for RNNs and LSTMs. As we use the last hidden state of RNNs for classification, this observation suggests that we need to use

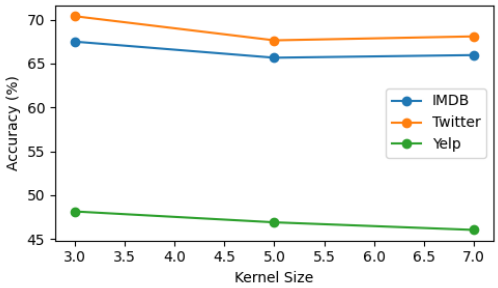


Figure 4: Accuracy vs. CNN kernel size (word embedding size = 50, hidden state size = 512).

the hidden state of the last token in the input text, and RNNs cannot preserve information when there are many padding tokens after the input text.

5 Conclusion

In this project, we evaluate CNNs, naive RNNs, and LSTMs on three sentiment analysis datasets. (1) We observe that CNNs and LSTMs show similar average performance, but each model performs better on different datasets. Naive RNNs are observed to be worse than both models in sentiment analysis. (2) We compare models with different hyperparameters (e.g., size of hidden states) but observe no clear trends, and larger models are not always better on the datasets we evaluate. (3) We also evaluate their performance on input texts with different lengths, but we observe different trends for different datasets. This result suggests that no simple factor influences their performance on sentiment analysis, and a deeper understanding of each dataset is required.

6 Lessons & Experience Learned in This Project

We should check raw data. We learned that it is important to carefully check the raw data before using it in machine learning methods. We initially thought that the Twitter dataset included three labels (positive, negative, and neutral), but it turned out to be a binary-class dataset. It took some time to notice this problem, as we did not check the raw data before feeding it to our training code.

We should make a detailed experiment plan. We learned that we should make a plan for experiments before starting large-scale training. As we did not make a detailed plan for our analysis, we needed to run experiments multiple times, and it took much more time than we expected. We could shorten the time for experiments if we had a clearer plan for our analysis.

References

[1] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[2] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. Technical report, Stanford University, 2009. CS224N Project Report.

[3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[4] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86:2278–2324, 1998.

[5] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In Dekang Lin, Yuji Matsumoto, and Rada Mihalcea, editors, *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[6] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In Kevin Knight, Hwee Tou Ng, and Kemal Oflazer, editors, *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[7] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 79–86. Association for Computational Linguistics, July 2002.

[8] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[9] Peter Turney. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In Pierre Isabelle, Eugene Charniak, and Dekang Lin, editors, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 417–424, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[10] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*, 2017.

[11] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.