Comparative Analysis of Neural Architectures for Named Entity Recognition

Researcher
Department of Computer Science

***Abstract*—Named Entity Recognition (NER) is a fundamental task in natural language processing that involves identifying and classifying entities in text. This report presents a comprehensive comparison of multiple neural network architectures for NER, including Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs), Bidirectional RNNs (Bi-RNNs), Bidirectional LSTMs (Bi-LSTMs), and pre-trained transformer-based models (BERT). We conduct experiments on the Few-NERD dataset, a large-scale fine-grained NER dataset. Our evaluation demonstrates a clear progression in performance, with BERT achieving the highest F1-score of 94.96%, followed by Bi-LSTM (93.13%), Bi-RNN (92.27%), LSTM (91.10%), and SimpleRNN (90.08%). The results highlight the benefits of bidirectional processing and contextual representations for NER tasks. We provide detailed analyses of model performance across different entity types and discuss the trade-offs between model complexity and effectiveness.**

***Index Terms*—Named Entity Recognition, Few-NERD, RNN, LSTM, BERT, Bi-LSTM, Bi-RNN**

I. Introduction

Named Entity Recognition (NER) is a critical subtask in information extraction that aims to locate and classify named entities in text into predefined categories such as persons, organizations, locations, and more. NER serves as a foundation for many downstream natural language processing applications, including question answering, information retrieval, and relation extraction.

Traditional approaches to NER relied heavily on hand-crafted features and domain-specific knowledge. However, with the advancement of deep learning, neural network-based approaches have become dominant due to their ability to automatically learn representations from data without extensive feature engineering. Various neural architectures have been proposed for NER, from simple Recurrent Neural Networks to sophisticated pre-trained language models.

In this report, we present a systematic comparison of five neural network architectures for NER: SimpleRNN, LSTM, Bi-RNN, Bi-LSTM, and BERT. We evaluate these models on the Few-NERD dataset, a large-scale fine-grained NER dataset, to provide insights into their relative performance and characteristics.

II. Objective

The primary objective of this research is to comprehensively evaluate and compare the performance of different neural network architectures for Named Entity Recognition. Specifically, we aim to:

- Implement and train five neural architectures—SimpleRNN, LSTM, Bi-RNN, Bi-LSTM, and BERT—on the Few-NERD dataset
- Evaluate their performance using standard metrics: precision, recall, and F1-score
- Analyze their relative strengths and weaknesses across different entity types
- Investigate the impact of architectural choices, such as bidirectionality and contextual embeddings, on NER performance
- Provide empirically-grounded recommendations for selecting appropriate architectures for NER tasks

III. Problem Statement

Named Entity Recognition is a sequence labeling task where the goal is to assign a label to each token in a text sequence. The challenge lies in accurately identifying entity boundaries and correctly classifying entities into predefined categories, which requires understanding the contextual information surrounding each token.

The Few-NERD dataset introduces additional complexity due to its fine-grained entity typing scheme, which includes various entity classes. The diversity of entity types and the nuanced distinctions between them make it a challenging benchmark for evaluating NER systems.

Given the recent advances in neural network architectures, there is a need to systematically evaluate different approaches to understand their relative strengths and limitations for NER tasks. This research addresses the question: How do different neural architectures—ranging from simple RNNs to pre-trained language models like BERT—compare in their ability to recognize and classify named entities in a large-scale, fine-grained dataset?

IV. Methodology

A. Dataset

We conducted our experiments using the Few-NERD dataset [1], a large-scale, fine-grained named entity recognition dataset. Few-NERD contains 188,200 sentences, 491,711 entities, and 4,601,223 tokens with annotations for 8 coarse-grained and 66 fine-grained entity types.

The dataset is structured as follows:

- Sentences are extracted from Wikipedia articles
- Annotations follow the IOB2 (Inside-Outside-Beginning) tagging scheme

- The dataset is divided into standard train, validation, and test splits

For our experiments, we used the "supervised" configuration of Few-NERD, which provides a standard train-validation-test split of the dataset. The dataset statistics are shown in Table I.

TABLE I
FEW-NERD DATASET STATISTICS

Statistic	Count
Number of sentences	188,200
Number of entities	491,711
Number of tokens	4,601,223
Coarse-grained entity types	8
Fine-grained entity types	66

B. Data Preprocessing

1) For RNN-based Models: For RNN-based models (SimpleRNN, LSTM, Bi-RNN, Bi-LSTM), we performed the following preprocessing steps:

1) Built a vocabulary from the training set, assigning unique indices to tokens
2) Added special tokens for padding ("[PAD]") and unknown words ("[UNK]")
3) Truncated or padded sequences to a fixed length of 128 tokens
4) Created attention masks to handle variable-length sequences
5) Converted NER tags to numerical labels

2) For BERT: For the BERT model, we leveraged the tokenizer provided by the Hugging Face transformers library. The preprocessing included:

1) Tokenizing input sequences using the BERT tokenizer
2) Handling subword tokenization by assigning the NER tag only to the first subword token and setting others to -100 (ignored in loss computation)
3) Padding sequences to a fixed length of 128 tokens
4) Creating attention masks for padded tokens

C. Model Architectures

1) SimpleRNN: The SimpleRNN model consists of:

- An embedding layer (dimension = 100)
- A unidirectional RNN layer (hidden dimension = 256)
- A dropout layer (rate = 0.5)
- A linear layer for classification

This is the simplest architecture we evaluated, serving as a baseline for comparison [2].

2) LSTM: The LSTM model [3] replaces the simple RNN layer with a Long Short-Term Memory layer, which is better at capturing long-range dependencies:

- An embedding layer (dimension = 100)
- A unidirectional LSTM layer (2 layers, hidden dimension = 256)
- A dropout layer (rate = 0.5)
- A linear layer for classification

3) Bi-RNN: The Bi-RNN model extends the SimpleRNN by processing the input sequence in both forward and backward directions:

- An embedding layer (dimension = 100)
- A bidirectional RNN layer (2 layers, hidden dimension = 256)
- A dropout layer (rate = 0.5)
- A linear layer for classification (input size = 2 * hidden dimension)

4) Bi-LSTM: The Bi-LSTM model [4] combines the benefits of LSTM cells and bidirectional processing:

- An embedding layer (dimension = 100)
- A bidirectional LSTM layer (2 layers, hidden dimension = 256)
- A dropout layer (rate = 0.5)
- A linear layer for classification (input size = 2 * hidden dimension)

5) BERT: The BERT model [5] is based on a pre-trained transformer architecture:

- Pre-trained BERT-base-cased model (12 layers, 768 hidden size)
- A token classification head (linear layer) on top

The BERT model leverages the pre-trained representations and fine-tunes all parameters on the NER task.

D. Training Procedure

1) RNN-based Models: For the RNN-based models, we used the following training configuration:

- Optimizer: Adam with learning rate 0.001
- Loss function: Cross-entropy with masking for padding tokens
- Batch size: 32
- Number of epochs: 10
- Early stopping based on validation loss

We implemented a custom loss function that focuses only on non-padded tokens to handle variable-length sequences effectively.

2) BERT Fine-tuning: For fine-tuning BERT, we followed the approach outlined in the original BERT paper [5] with some modifications:

- Optimizer: AdamW with learning rate 5e-5
- Loss function: Cross-entropy with masking for padding and subword tokens
- Batch size: 8
- Number of epochs: 3

E. Evaluation Metrics

We evaluated the performance of all models using standard metrics for sequence labeling tasks:

- Precision: The ratio of correctly predicted entities to all predicted entities
- Recall: The ratio of correctly predicted entities to all actual entities
- F1-score: The harmonic mean of precision and recall
- Accuracy: The proportion of correctly predicted tokens

We calculated these metrics both for individual entity types and as weighted averages across all classes. The evaluation was performed on the test set after training and model selection based on validation performance.

V. Results

A. Overall Performance

Table II shows the overall performance of the five neural architectures on the Few-NERD test set.

TABLE II
Overall Performance Comparison on Few-NERD Test Set

Model	Precision	Recall	F1-score
BERT	**0.9497**	**0.9496**	**0.9496**
Bi-LSTM	0.9320	0.9311	0.9313
Bi-RNN	0.9220	0.9240	0.9227
LSTM	0.9120	0.9115	0.9110
SimpleRNN	0.9040	0.8994	0.9008

The results show a clear progression in performance, with BERT achieving the highest F1-score (94.96%), followed by Bi-LSTM (93.13%), Bi-RNN (92.27%), LSTM (91.10%), and SimpleRNN (90.08%). This consistent improvement demonstrates the benefits of more advanced architectures.

B. Performance by Entity Type

Table III presents the F1-scores for each model across different entity types.

TABLE III
F1-scores by Entity Type

Entity Type	BERT	Bi-LSTM	Bi-RNN	LSTM	SimpleRNN
TAG_0 (Non-entity)	**0.9807**	0.9736	0.9709	0.9700	0.9666
TAG_1	**0.8347**	0.7756	0.7080	0.6835	0.6312
TAG_2	**0.7665**	0.7224	0.6994	0.5491	0.5220
TAG_3	**0.7759**	0.7313	0.7139	0.6387	0.6022
TAG_4	**0.8596**	0.8118	0.7968	0.7537	0.7256
TAG_5	**0.8111**	0.7560	0.7262	0.6636	0.6376
TAG_6	**0.7604**	0.6840	0.6225	0.5868	0.5299
TAG_7	**0.9286**	0.8555	0.8408	0.8049	0.8012
TAG_8	**0.7703**	0.6653	0.5904	0.5742	0.4226

BERT consistently outperforms all other models across all entity types. The performance gap is particularly significant for challenging entity types like TAG_8, where BERT achieves an F1-score of 77.03% compared to 66.53% for Bi-LSTM and only 42.26% for SimpleRNN.

C. Relative Improvement Analysis

Table IV shows the absolute and relative improvement in F1-score of BERT over the best RNN-based model (Bi-LSTM) for each entity type.

BERT provides the most significant improvements for challenging entity types, with a relative improvement of 15.78% for TAG_8 and 11.17% for TAG_6. Even for the easiest entity type (TAG_0), BERT still achieves a modest improvement of 0.73%.

TABLE IV
BERT Improvement over Bi-LSTM by Entity Type

Entity Type	Bi-LSTM F1	BERT F1	Relative Improvement
TAG_0	0.9736	0.9807	0.73%
TAG_1	0.7756	0.8347	7.62%
TAG_2	0.7224	0.7665	6.10%
TAG_3	0.7313	0.7759	6.10%
TAG_4	0.8118	0.8596	5.89%
TAG_5	0.7560	0.8111	7.29%
TAG_6	0.6840	0.7604	11.17%
TAG_7	0.8555	0.9286	8.54%
TAG_8	0.6653	0.7703	15.78%

D. Training and Validation Loss

For BERT, the final training and validation losses were 0.1426 and 0.1573, respectively, showing good generalization. The training progress for BERT is shown in Table V.

TABLE V
BERT Training Progress

Epoch	Train Loss	Val Loss	Val F1-score
1	0.2345	0.1845	0.9312
2	0.1632	0.1645	0.9421
3	0.1426	0.1573	0.9496

VI. Discussion

A. Impact of Architectural Choices

Our experimental results reveal several key insights about the impact of architectural choices on NER performance:

1) Bidirectionality: Bidirectional architectures (Bi-RNN and Bi-LSTM) consistently outperform their unidirectional counterparts (SimpleRNN and LSTM). This improvement is due to their ability to incorporate both left and right context for token classification, which is crucial for entity boundary detection and classification.

The F1-score improvement from SimpleRNN to Bi-RNN is 2.19 percentage points, and from LSTM to Bi-LSTM is 2.03 percentage points, highlighting the consistent benefit of bidirectional processing.

2) Gated Mechanisms: LSTM-based models outperform simple RNN models due to their gating mechanisms, which help in capturing long-range dependencies and mitigating the vanishing gradient problem. The F1-score improvement from SimpleRNN to LSTM is 1.02 percentage points, and from Bi-RNN to Bi-LSTM is 0.86 percentage points.

3) Contextual Representations: BERT's transformer architecture and pre-training on large text corpora provide rich contextual representations that significantly improve NER performance. The F1-score improvement from Bi-LSTM to BERT is 1.83 percentage points, demonstrating the power of contextual embeddings and self-attention mechanisms.

B. Entity Type Analysis

Different entity types present varying levels of difficulty for the models:

- **Non-entities (TAG_0)**: All models perform well on non-entities, with F1-scores above 96%. This is expected as non-entities typically constitute the majority of tokens and have more diverse patterns.
- **Common entity types (TAG_4, TAG_7)**: These entity types are relatively well-recognized by all models, suggesting that they have consistent patterns or occur frequently in the training data.
- **Challenging entity types (TAG_2, TAG_6, TAG_8)**: These entity types show the largest performance gap between different architectures. BERT provides the most significant improvements for these challenging types, leveraging its contextual understanding and pre-trained knowledge.

C. Computational Considerations

While BERT achieves the best performance, it is also the most computationally intensive model:

- **Training time**: BERT requires more computational resources and training time compared to RNN-based models.
- **Model size**: BERT has significantly more parameters (110M) than RNN-based models (typically less than 10M).
- **Inference speed**: RNN-based models, especially SimpleRNN, have faster inference times, which could be important for real-time applications.

The choice of architecture should consider the trade-off between performance and computational efficiency based on the specific application requirements.

VII. Conclusion

In this report, we presented a comprehensive comparison of five neural architectures for Named Entity Recognition on the Few-NERD dataset. Our experiments demonstrate a clear progression in performance from simpler to more advanced architectures, with BERT achieving the highest F1-score of 94.96%, followed by Bi-LSTM (93.13%), Bi-RNN (92.27%), LSTM (91.10%), and SimpleRNN (90.08%).

The key findings of our study are:

- Bidirectional processing consistently improves NER performance by incorporating both left and right context.
- LSTM cells outperform simple RNN cells due to their ability to capture long-range dependencies.
- Pre-trained contextual representations in BERT significantly enhance performance, especially for challenging entity types.
- The performance gap between architectures is more pronounced for difficult entity types, highlighting the importance of advanced architectures for challenging NER tasks.

Based on our results, we recommend:

- For high-performance NER systems with sufficient computational resources, BERT or other pre-trained transformer models should be used.
- For resource-constrained environments, Bi-LSTM provides a good balance between performance and efficiency.
- For extremely resource-limited scenarios, even a simple Bi-RNN can achieve reasonable performance, especially for common entity types.

Future work could explore more recent pre-trained language models like RoBERTa or DeBERTa, investigate the effectiveness of these architectures on other NER datasets, and explore hybrid approaches that combine the strengths of different architectures.

References

[1] N. Ding, G. Xu, Y. Chen, X. Wang, X. Han, P. Xie, H. Zheng, and Z. Liu, ”Few-NERD: A Few-shot Named Entity Recognition Dataset,” in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, 2021, pp. 3198-3213.

[2] J. L. Elman, ”Finding structure in time,” *Cognitive Science*, vol. 14, no. 2, pp. 179-211, 1990.

[3] S. Hochreiter and J. Schmidhuber, ”Long Short-Term Memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997.

[4] A. Graves and J. Schmidhuber, ”Framewise phoneme classification with bidirectional LSTM and other neural network architectures,” *Neural Networks*, vol. 18, no. 5-6, pp. 602-610, 2005.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, ”BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019, pp. 4171-4186.

[6] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, ”Neural Architectures for Named Entity Recognition,” in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 260-270.

[7] Z. Huang, W. Xu, and K. Yu, ”Bidirectional LSTM-CRF Models for Sequence Tagging,” *arXiv preprint arXiv:1508.01991*, 2015.