Course Project

GRUNER: GRU-based Named Entity Recognition Model

Project Report

Dev Patel
Hasnat Alam
Ravi Pandey

College of Information Science & Technology
University of Nebraska at Omaha

Abstract

Named Entity Recognition (NER) is a key task in Natural Language Processing (NLP), and it is particularly critical in specialized domains like biomedicine. This project investigates the use of a Gated Recurrent Unit (GRU)-based deep learning model for NER on the Colorado Richly Annotated Full-Text (CRAFT) Corpus [1], which provides rich biomedical text annotations. The CRAFT data was first preprocessed into the Inside-Outside-Beginning (IOB) format to label tokens as Gene Ontotlogy (GO) term entities or not using the annotations in the dataset for three GO categories: Molecular Function, Biological Process, and Cellular Component. Then the obtained word-tag dataset was used to build and train a Gated Recurrent Unit (GRU) based NER model. Domain-specific embeddings were generated using BioBERT to enhance semantic representation of biomedical terms. Various hyperparameter configurations, including different hidden dimensions and learning rates, were evaluated to optimize model performance. Precision, Recall and F1 scores were used to evaluate the model performance and results indicate that a GRU model with high number of hidden dimensions and a learning rate of 0.09 achieved the best performance. The study highlights the trade-offs between precision and recall and emphasizes the critical role of hyperparameter tuning. This project demonstrates that GRU-based models can be trained for domain-specific tasks such as biomedical NER to obtain satisfactory performance.

Table of Contents

1. Introduction

1.1. Overview and Motivation

Named Entity Recognition (NER) is a foundational task in Natural Language Processing (NLP) that involves identifying and categorizing entities such as names of people, organizations, and locations within text [2]. It is a subfield of information extraction and plays a key role in structuring unstructured text data [3]. In the biomedical domain, NER is particularly critical for identifying entities such as genes, proteins, diseases, and chemicals, supporting information retrieval, knowledge discovery, and research advancement [4]. Biomedical NER (BioNER) poses unique challenges due to domain-specific terminology, abbreviations, and the continuous expansion of biomedical vocabularies [4, 5].

This project focuses on building an NER pipeline using a Bi-GRU-based model trained on the Colorado Richly Annotated Full-Text (CRAFT) Corpus [1], progressing through two phases: (1) preprocessing gold standard data and applying IOB tagging, and (2) creating custom word embeddings and training the sequence model. The motivation is to study how different preprocessing strategies and model configurations influence NER performance in specialized biomedical texts.Incorporating domain knowledge and contextual modeling has been shown to significantly improve NER accuracy in biomedical applications [5].

1.2. Scope and Focus of the Report

This project focuses on the development of a NER model tailored for biomedical text, using the CRAFT corpus as the primary dataset. The scope of the project includes preprocessing annotated texts into a machine-learning-friendly format through tokenization and Inside-Outside-Beginning (IOB) tagging [6], followed by creating word embeddings and training a Bi-GRU-based sequence model [5]. It also focuses on systematically experimenting with different hyperparameter settings, such as hidden layer dimensions and learning rates, to analyze their effect on model performance. The report highlights design decisions, training procedures, evaluation metrics, and results analysis aimed at identifying configurations that optimize precision, recall, and F1 score for biomedical entity recognition.

2. Methodology

2.1. Dataset

The dataset used in this project is derived from the CRAFT corpus, a widely recognized resource in biomedical NLP. The CRAFT corpus provides full-text biomedical journal articles richly annotated with various biomedical ontologies. For this project, annotations related to Gene Ontology (GO) terms were utilized. The annotations are categorized into three main types:

- GO_MF (Molecular Function)
- GO_BP (Biological Process)
- GO_CC (Cellular Component)

In the repository, each article's raw text is stored in `articles/txt/[ID].txt`, while the corresponding annotations are stored in ontology-specific subdirectories in xml format. Each XML file contains multiple `<annotation>` elements inside `<annotations>`. Within an annotation, one or more `<span start="..." end="..."/>` tags indicate character offsets (begin and end) of the annotated text, and a single `<spannedText>` tag provides the exact string covered by those spans. An example annotation with discontinuous spans is shown below:

```
<annotation>
    <mention id="GO_BP_2016_02_16_Instance_28421" />
    <annotator id="GO_BP_2016_02_16_Instance_10000">Mike Bada,
    University of Colorado Anschutz Medical Campus</annotator>
    <span start="2556" end="2568" />
    <span start="2581" end="2590" />
    <spannedText>formation of ... complexes</spannedText>
  </annotation>
Note: Here, the "..." indicates a discontinuous annotation.
```

To prepare the data for sequence labeling tasks like Named Entity Recognition (NER), the annotations were converted into the Inside-Outside-Beginning (IOB) tagging format, where each token is labeled to indicate whether it is inside, outside, or at the beginning of an entity.

2.2. Data Preprocessing

The preprocessing pipeline transforms each raw article and its XML annotations into a sentence-level list of `(word,tag)` pairs in IOB format. The main steps are:

1. **Annotation Extraction:** For each ontology XML file, we parse every `<annotation>` element and collect its spans in a list of 4-tuples:

$$(\text{start}, \text{end}, \text{spannedText}, \text{nextStart}),$$

   where nextStart is the start offset of a second span if the annotation is discontinuous (or 0 otherwise). This representation helps preserve discontinuous entities when tagging.

2. **Tokenization and Alignment:** We load the raw text from `articles/[ID].txt` and tokenize it with `spaCy`. For each token, we compute its character-based start and end positions and match these against our span tuples. Tokens falling entirely within a span are labeled "B" for the first token in the span (IOB "B") and "I" for subsequent tokens. All other tokens are labeled "O."

   When a discontinuous annotation is encountered, the words between it's tagged components are dropped from the final result.

3. **Overlap Resolution:** When two annotations overlap in their character offsets, we discard the smaller span (shorter character length) to avoid conflicting labels.

4. **Output:** For each input file, after tagging, we produce a corresponding output file containing a Python-style list of sentences, where each sentence is itself a list of `(word, tag)` tuples.

This structured preprocessing ensures accurate token-level IOB labels, accommodates discontinuous and overlapping annotations, and outputs data in a consistent format for subsequent embedding generation and model training.

2.2.1 Training and validation split

Data is randomly split 80% for training and 20% for validation before training the model.

2.2.2 Data normalization or filtering

Models are trained on the entire dataset, and then we filter out the data set based on I and B tags in any sentence. Two filters are used one based on four number of tags and second filter out data based on eight number of tags in any sentence. This filtering is done to improve the class imbalance since the original data contains a large number of tokens tagged as 'O' as compared to tokens tagged with both 'B' and 'I' tags [7].

2.3. Model Architecture

. The NER model is built on a bidirectional gated repeat unit (GRU)-based sequence labeling architecture. Two activation variants were evaluated to improve training stability and gradient flow. The architecture comprises:

- **Embedding Layer**:

  Maps each input token to a dense vector representation. This layer captures semantic relationships between words based on their contextual usage in the corpus.

- **Bidirectionnal GRU Layer**:

  This layer processes the embeddings in both forward and backward directions. The hidden size is set by a hyperparameter, allowing the model to learn complex sequential dependencies.

- **Activation Variants**:

  Two versions of the GRU output were tested:

  - *Adam-activated GRU*: uses the standard GRU recurrence with the Adam optimizer for parameter updates.
  - *ReLU-activated GRU*: applies a ReLU nonlinearity to the GRU outputs before the final layer, enhancing gradient flow and improving convergence stability.

- **Linear Output Layer**:

  Projects the bidirectional GRU hidden states to the IOB tag space, producing token-level classification logits.

The simplicity of the architecture was intended to allow a focused investigation into the ef-

fects of preprocessing and hyperparameter settings without introducing additional complexity.

2.4. Hyperparameter Settings

To optimize the performance of the model, various hyperparameter configurations were tested systematically through a cpmprehensive grid search. These settings were chosen to evaluate their impact on the model's learning efficiency and performance. The following hyperparameters were explored:

- **Hidden Dimensions**: {64, 128, 256, 512, **1024, 2048, 4096**}

  These values correspond to the size of the hidden state vectors in the Gated Recurrent Unit (GRU) layer. Larger hidden dimensions allow the model to capture more complex patterns and dependencies in the input sequences, but may also increase the risk of overfitting.

- **Learning Rates**: {0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, **0.09**}

  The learning rate controls the step size during the optimization process. A lower learning rate ensures more gradual convergence but can also lead to slower training, while a higher learning rate accelerates convergence, but risks overshooting the optimal solution.

- **Batch Size**: {64, 128, 256}

  Batch size means the number of samples processed before updating the model parameters, balancing memory usage and gradient estimate quality.

- **Number of Epochs**: {1000, 2000, **3000**}

  The number of epochs defines how many times the entire training data set is passed through the model. A higher number of epochs allows for more thorough training, which could improve the accuracy of the model, but too many epochs can lead to overfitting if not properly monitored.

- **Optimizers**:

  - **Adagrad**: An adaptive learning rate optimizer that adjusts the learning rate based on the parameters' past gradients, making it effective for sparse data.
  - **Adam**: An adaptive optimizer that combines the benefits of both Adagrad and RMSProp, using moving averages of gradients and their squares to adjust learning rate.
  - **SGD (Stochastic Gradient Descent)**: A basic optimization method that updates

the model parameters using the gradient of the loss function with respect to the parameters, often with a fixed learning rate.

Initially, the model was trained with a lower number of hidden dimensions (64 to 512), but the evaluation metric scores were not as desired, so we increased the number of hidden dimensions and trained the model from 1024 to 4096 hidden dimensions.

Similarly, we did not see a satisfactory metric score for lower learning rates, so we also trained the model with a high learning rate of 0.09. These changes, coupled with the larger number of epochs (3000), helped improve model performance.

The goal of this grid search was to identify the optimal combination of hyperparameters that would enhance the performance of the model on unseen data, balancing both training efficiency and generalization.

2.5. Evaluation Metrics

Model performance was evaluated using standard metrics commonly applied in sequence labeling and Named Entity Recognition (NER) tasks. Precision was used to measure the proportion of correctly predicted positive labels out of all labels predicted as positive. Recall, on the other hand, measured the proportion of correctly predicted positive labels out of all actual positive labels in the data. To provide a single comprehensive measure that balances both precision and recall, the F1-Score was calculated. The F1-Score is the harmonic mean of precision and recall, providing a balanced metric that is especially useful when there is an uneven class distribution or when both precision and recall are important. These metrics were computed for each model variant on the validation and test sets, and they served as the basis for selecting the best-performing model configurations.

We computed two sets of evaluation metrics for each training configuration:

- **Actual tags (I, B, O)**: Evaluation metrics (precision, recall, F1 score) were calculated for each individual tag, considering tokens labeled as Inside (I), Beginning (B), or Outside (O) of an entity.
- **Grouped tags (IB, O)**: Evaluation metrics were recalculated using a binary classification where Inside (I) and Beginning (B) tags are grouped as a single entity class (IB) or Non-O and all Outside (O) tokens are treated as the non-entity class.

This allows comparison of performance both on fine-grained entity classification and on distinguishing entities from non-entities.

3. Results

After completing model training across various hyperparameter configurations, we evaluated the models to assess their performance on the biomedical Named Entity Recognition (NER) task. This section presents the experimental results, focusing on the impact of different hidden dimensions, learning rates, and training durations on precision, recall, and F1 score.

Table 1 presents the performance of the models under various configurations for 2 training runs. The experiments involved testing multiple values for hidden dimensions, learning rates, and the number of epochs to determine their effects on the model's ability to recognize named entities in the biomedical domain.

Table 1: GRUNER evaluation results for Run 1 (Epochs=3000, Batch size=64, Optimizer=Adam). Metrics reported are Precision, Recall, and F1 Score.

(a) All

Hidden Dim	LR	Precision	Recall	F1
1024	0.01	0.6667	0.2491	0.3608
1024	0.05	0.4381	0.1350	0.2063
1024	0.09	0.4074	0.2201	0.2857
2048	0.01	0.4405	0.1703	0.2450
2048	0.05	0.5093	0.1639	0.2458
2048	0.09	0.4646	0.1993	0.2781
4096	0.01	0.5417	0.2264	0.3172
4096	0.05	0.5167	0.2364	0.3105
4096	0.09	0.4771	0.2717	0.3379

(b) O and No-O

Hidden Dim	LR	Precision	Recall	F1
1024	0.01	0.5000	0.1923	0.2778
1024	0.05	0.5000	0.1538	0.2353
1024	0.09	0.5000	0.2692	0.3500
2048	0.01	0.5000	0.1923	0.2778
2048	0.05	0.5000	0.1667	0.2500
2048	0.09	0.5000	0.2179	0.3036
4096	0.01	0.5000	0.2051	0.2909
4096	0.05	0.5000	0.2436	0.3276
4096	0.09	0.5000	0.2949	0.3710

Table 2: GRUNER evaluation results for Run 2 (Epochs=3000, Batch size=64, Optimizer=Adam). Metrics reported are Precision, Recall, and F1 Score.

(a) All

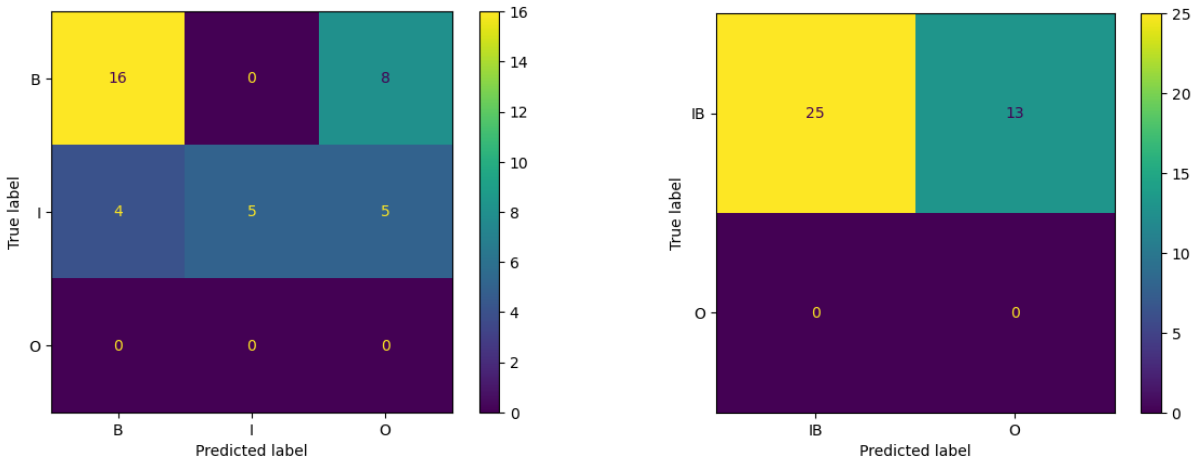
Hidden Dim	LR	Precision	Recall	F1
1024	0.01	0.6000	0.2857	0.3806
1024	0.05	0.5598	0.2718	0.3649
1024	0.09	0.5439	0.2143	0.2694
2048	0.01	0.6111	0.2579	0.3606
2048	0.05	0.5397	0.2718	0.3596
2048	0.09	0.4667	0.2540	0.3235
4096	0.01	0.5737	0.3194	0.4103
4096	0.05	0.4469	0.2341	0.3072
4096	0.09	0.6000	0.3413	0.4179

(b) O and No-O

Hidden Dim	LR	Precision	Recall	F1
1024	0.01	0.5000	0.2632	0.3448
1024	0.05	0.5000	0.2500	0.3333
1024	0.09	0.5000	0.2763	0.3559
2048	0.01	0.5000	0.2237	0.3091
2048	0.05	0.5000	0.2632	0.3448
2048	0.09	0.5000	0.2632	0.3448
4096	0.01	0.5000	0.2763	0.3559
4096	0.05	0.5000	0.2632	0.3448
4096	0.09	0.5000	0.3289	0.3968

The best performance was achieved with a configuration of 4096 hidden dimensions and a learning rate of 0.09, resulting in a precision of 0.5, recall of 0.33 and F1 Score of ~ 0.40 when measured for 3 distinct classes. Other configurations showed varying levels of performance, with some models achieving similar precision but lower recall, while others displayed lower overall performance.

The contingency matrix for the best performance model configuration is shown below in figure 1:

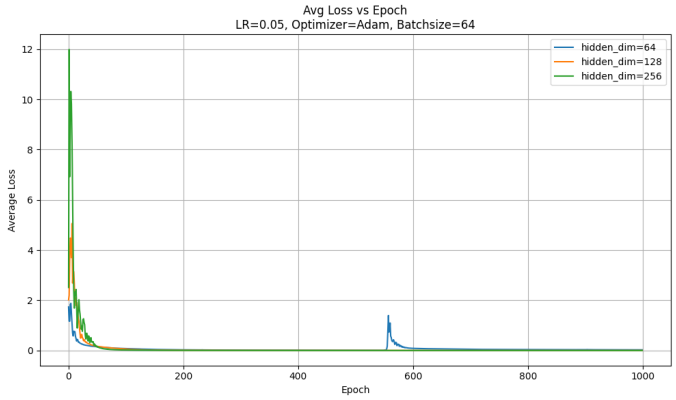


(a) I, O and B classes

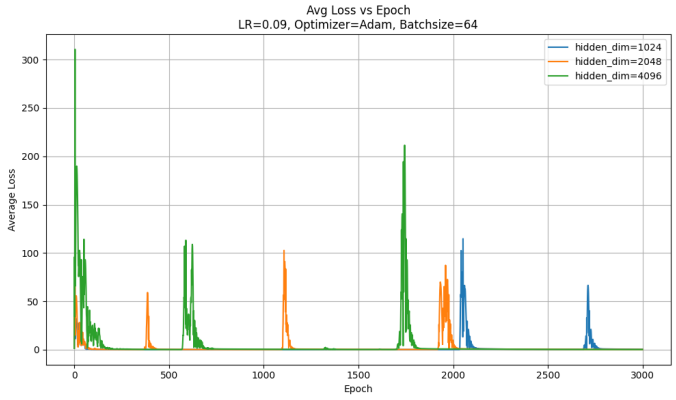
(b) O and Non-O (IB) classes

Figure 1: Contigency matricies for best performing model configuration

The loss function for the training loop has been visualized for 1000 epochs and 3000 epochs in figures 2a and 2b respectively.



(a) 1000 Epochs



(b) 3000 Epochs

Figure 2: Loss Functions

4. Comparative Analysis

The model performance across different hyperparameter configurations reveals several trends and trade-offs that are crucial to understanding the impact of each parameter on the biomedical NER task.

4.1. Hyperparameter Impact

- **Learning Rates**: The choice of learning rate had a significant effect on model convergence and recall. Lower rates (0.01) produced slow convergence with modest recall gains (e.g. recall ≈ 0.1923 at 1024 hidden, Adam), while higher rates (0.09) accelerated training and significantly increased recall (recall ≈ 0.264 at 1024 hidden, Adam). However, excessively high rates risked instability without ReLU; When combined with ReLU activation, even the highest rate (0.09) produced smoother learning curves and a higher final recall (0.2763) and F1 (0.3559), demonstrating that an aggressive learning rate can be harnessed effectively if the gradient flow is well conditioned.
- **Hidden Dimensions**: Increasing hidden dimensions from 1024 to 4096 improved the model's capacity to capture complex biomedical contexts, reflected in the increased recall (from 0.2692 to 0.2949 at LR 0.09, Adam) and F1 (from 0.3500 to 0.3710). The addition of ReLU further amplified these gains (F1 up to 0.3968 at 4096, LR 0.09). The marginal F1 improvements beyond 2048 hidden units indicate diminishing returns: doubling capacity yields only incremental performance boosts and can introduce slight precision instability (e.g. precision drop to 0.4667 at 2048, LR 0.09, Adam+ReLU).

4.2. Trade-offs

A key trade-off observed is between precision stability and recall-driven F1 improvements. Precision was stable around 0.5000 across nearly all settings, with only one small drop, while recall was the primary driver of F1 gains. Higher learning rates and larger hidden sizes improved recall, but required LU to avoid training oscillations. Thus, achieving high F1 requires balancing aggressive hyperparameters with activation schemes that stabilize gradient updates.

4.3. General Observations

- Models with lower hidden dimensions performed poorly, probably because the model lacked the capacity to capture the complexities of biomedical text. This resulted in very low precision and recall.
- The high number of epochs was crucial in stabilizing model training. In future experiments, further tuning of the epoch numbers could yield better model performance, especially for larger models.

5. Conclusion

This project demonstrated a complete pipeline for biomedical Named Entity Recognition (NER) on the CRAFT corpus. First, the raw CRAFT XML annotations were converted into token-level IOB format, and, the data were enriched with BioBERT embeddings, which produced a high-quality training dataset. Next, we designed and implemented a GRU-based bidirectional sequence labeling model, experimenting with standard Adam updates and a Adam and ReLU activation variant. A comprehensive search of the grid for hidden dimensions, learning rates, batch size, and 3000 training epochs identified the best performing configuration. The metrics showed that while high precision was maintained, recall improvements, especially when using ReLU, drove overall F1 gains. F1 score is high when all data are used without any filtering but with low precision. By increasing the number of hidden layers, the precision and F1 score improves. RELU activation function helps to improve the model accuracy, but the improvements are not very significant. The model accuracy improved with small batch size, high learning rate, and high number of epcohs. These results underscore the importance of careful hyperparameter tuning and activation choices in GRU-based NER. In general, the project results suggest that combining BioBERT embeddings with a well-tuned GRU architecture is a practical and effective approach to biomedical entity recognition, with the potential to be further optimized for specific use cases.

References

[1] K. B. Cohen, K. Verspoor, K. Fort, C. Funk, M. Bada, M. Palmer, and L. E. Hunter, "The colorado richly annotated full text (craft) corpus: Multi-model annotation in the biomedical domain," in *Handbook of Linguistic Annotation*, N. Ide and J. Pustejovsky, Eds. Springer, 2017, pp. 1379–1394. [Online]. Available: https://doi.org/10.1007/978-94-024-0881-2_55

[2] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*, 3rd ed., 2025, online manuscript released January 12, 2025. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/

[3] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification," *Lingvisticæ Investigationes*, vol. 30, no. 1, pp. 3–26, 2007.

[4] J.-D. Li, Y. Sun, R. J. Johnson, D. Sciaky, C.-H. Wei, R. Leaman, A. P. Davis, C. J. Mattingly, T. C. Wiegers, and Z. Lu, "Biocreative v cdr task corpus: a resource for chemical disease relation extraction," *Database: The Journal of Biological Databases and Curation*, vol. 2016, p. baw068, 2016.

[5] H. Cho and H. Lee, "Biomedical named entity recognition using deep neural networks with contextual information," *BMC Bioinformatics*, vol. 20, no. 1, p. 735, 2019. [Online]. Available: https://doi.org/10.1186/s12859-019-3321-4

[6] L. A. Ramshaw and M. P. Marcus, "Text chunking using transformation-based learning," in *Third Workshop on Very Large Corpora*, 1995, available at: https://aclanthology.org/W95-0107. [Online]. Available: https://aclanthology.org/W95-0107

[7] M. Ali *et al.*, "Sentence-based undersampling for named entity recognition using genetic algorithm," *Iran Journal of Computer Science*, vol. 1, no. 3, pp. 165–174, 2018. [Online]. Available: https://link.springer.com/article/10.1007/s42044-018-0014-5