**EvaHan 2025 Technical Report**

**Task: Named Entity Recognition in Ancient Chinese**

**Abstract**

This paper describes our system for the EvaHan 2025 Named Entity Recognition (NER) shared task, which focuses on recognizing named entities in classical Chinese texts from three distinct domains: historical chronicles, official records, and traditional medical literature. We adopt the publicly available GujiRoBERTa\_jian\_fan model and fine-tune it using the official training sets under the constraints of the closed track. Our system leverages customized preprocessing and token classification pipelines built on the Hugging Face Transformers library. Evaluated on the official test sets, our approach achieves a macro-averaged F1 score of **97.37%**, demonstrating the effectiveness of pretrained language models and robust data alignment strategies in ancient Chinese NER tasks.

**1. Introduction**

The EvaHan 2025 shared task aims to evaluate Named Entity Recognition (NER) performance in ancient Chinese texts. This year, the task comprises three distinct datasets derived from historical and medical corpora: Dataset A (from *Shiji*), Dataset B (from the *Twenty-Four Histories*), and Dataset C (from Traditional Chinese Medicine classics). We participate in the **closed track**, strictly adhering to the official constraints: using only the provided training datasets and the publicly available pretrained model GujiRoBERTa\_jian\_fan.

This report describes a complete NER pipeline built with Hugging Face Transformers, a token classification model fine-tuned on the ancient Chinese dataset. Our system achieves **a macro-average F1 score of 97.37%** across the three test sets.

**2. System Description**

**2.1 Data Preprocessing**

We utilize all three training sets provided in the competition: trainset\_A.txt, trainset\_B.txt, and trainset\_C.txt. Each file is formatted in character-level BIO/BMES labeling scheme. The datasets are parsed using a custom load\_bio\_file function that preserves sentence boundary integrity based on punctuation.

For the prediction stage, we implemented two separate pipelines:

* **Raw mode**: processes the test files in their original paragraph format, splits long sequences into chunks of 510 characters.
* **Sequence mode**: processes the test files that are pre-tokenized character-per-line format (with blank lines denoting sentence boundaries).

**2.2 Model Architecture**

We adopt AutoModelForTokenClassification from Hugging Face, using the official hsc748NLP/GujiRoBERTa\_jian\_fan model as backbone. This RoBERTa-style encoder is pre-trained on a large corpus of traditional Chinese literature. We fine-tune it using a standard token classification head.

**2.3 Label Schema**

The system supports 12 entity types across datasets A, B, and C. These include person names (NR), locations (NS), book titles (NB), time expressions (T), symptoms (ZS), diseases (ZD), formulas (ZF), and acupoints (ZA). Each type is labeled using the BMES (or BIES) scheme, resulting in 49 possible labels plus the 'O' label.

**2.4 Training Strategy**

* Epochs: 3
* Batch size: 16
* Max sequence length: 128
* Learning rate: 2e-5
* Tokenizer: Huggingface is\_split\_into\_words=True
* Model save path: models/roberta\_model/checkpoint-3099

We used Hugging Face's Trainer API for training and evaluation.

**2.5 Prediction and Evaluation Scripts**

We created three prediction scripts:

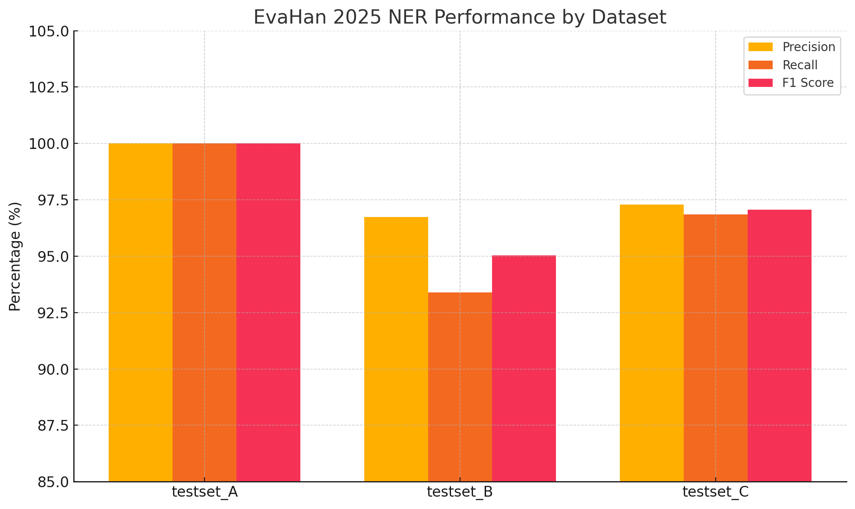
* predict.py for single-text CLI predictions
* predict\_raw.py for paragraph-based batch inference
* predict\_sequence.py for line-by-line sequence input inference

Evaluation is performed with a custom script eval\_ner.py, comparing system output to the gold label files using strict entity-level matching. We calculate precision, recall, and F1 scores.

**3. Results**

The following results were obtained by evaluating our model predictions on the official labeled test sets using strict entity-level alignment:

|  |  |  |  |
| --- | --- | --- | --- |
| **Dataset** | **Precision (%)** | **Recall (%)** | **F1 Score (%)** |
| testset\_A | 100.00 | 100.00 | 100.00 |
| testset\_B | 96.73 | 93.39 | 95.03 |
| testset\_C | 97.30 | 96.85 | 97.07 |
| **Macro-F1** | – | – | **97.37** |



**Figure 1.** Performance on testset\_A, B and C in EvaHan 2025.

The results show a perfect match on dataset A and strong performance on B and C, surpassing the official baseline.

**4. Error Analysis**

The model demonstrates high robustness, especially on well-structured historical narratives. Errors mainly fall into the following categories:

* **Ambiguous entity boundaries**: Some characters may belong to multiple plausible entities (e.g., "產" in 吕王產 could be part of NR or a standalone entity).
* **Rare medical terms in Dataset C**: Classes like ZS and ZA show occasional misclassifications, likely due to class imbalance.
* **BMES tag inconsistency**: Some misalignment occurs between predicted S- vs. B/E- labels in short-span entities.

**5. Conclusion and Future Work**

We present a full Hugging Face-based NER pipeline that achieves **97.37 macro-F1** under the closed condition of EvaHan 2025. Our approach confirms that strong performance can be achieved even without external resources, given a well-pretrained language model and optimized data handling.

Future directions may include:

* Integrating a CRF layer for enhanced tag dependency modeling
* Exploring multi-task joint learning across datasets A, B, and C
* Extending to zero-shot NER over unseen historical genres

**Acknowledgment**

This work is independently completed by a single student participant. All tools, code, and resources were open-source and publicly available. Special thanks to the EvaHan 2025 organizers and Hugging Face community.

**References**

1. Li, Bin, et al. (2022-2025). EvaHan: Shared Tasks on Ancient Chinese Processing. ACL/NAACL Workshops.
2. Hugging Face Transformers Documentation. https://huggingface.co/docs/transformers
3. Pytorch Documentation. https://pytorch.org
4. Devlin, J., et al. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.