Reproducibility Study for Rethinking Negative Instances for Generative Named Entity Recognition

Mamnuya Rinki
George Mason University
mrinki@gmu.edu

Mary Rithika Reddy Gade
George Mason University
mgade2@gmu.edu

1 Introduction

We explore the reproducibility of Rethinking Negative Instances for Generative Named Entity Recognition (GNER) (Ding et al., 2024). This paper claims there is enhanced zero-shot capabilities for unseen entity domains using GNER-LLaMA and GNER-T5. They found improvements of 8 and 11 points in F-1 score, respectively, when compared to LLaMA and Flan-T5.

1.1 Task / Research Question Description

The paper aims to address the challenge of improving zero-shot named entity recognition (NER) in generative models by rethinking the treatment of negative instances text that does not contain named entities. Specifically, it seeks to answer whether incorporating non-entity text as negative instances can enhance a model's performance in zero-shot scenarios, enabling it to identify named entities in unseen domains effectively. The primary research question is: *Can generative models, trained with better handling of negative instances, improve F1 scores in zero-shot entity recognition tasks across diverse domains?*

1.2 Motivation & Limitations of Existing Work

Previous work in zero-shot NER has largely focused on GPT-based models like GPT-3 and ChatGPT, which excel in zero-shot settings but often lack robust handling of non-entity text, leading to classification errors. Other approaches, such as GoLLIE (Sainz et al., 2024), have broadened their NER datasets to improve domain coverage but have not fully addressed the impact of non-entity text in generative models. These models show promise in entity recognition across domains but fall short in accurately distinguishing between entity and non-entity text.

The motivation behind this study is to leverage generative models (GNER-LLaMA and GNER-T5) that are trained with non-entity text as negative instances to better distinguish between entities and non-entities, aiming for a more precise zero-shot NER system. The limitation of prior work lies in their reliance on annotated entity-focused datasets and lack of non-entity handling, which limits generalizability and accuracy across unseen domains.

1.3 Proposed Approach

The core contribution of the paper is the introduction of the Generative NER (GNER) framework, which trains generative models like LLaMA and Flan-T5 with carefully selected negative instances (non-entity text) alongside entity-labeled data. This approach enables the models to develop a clearer understanding of what constitutes a non-entity, which improves their zero-shot performance. By incorporating these negative instances, GNER models (GNER-LLaMA and GNER-T5) enhance their zero-shot NER capabilities, achieving higher F1 scores compared to traditional models, and demonstrating better performance in distinguishing entities from non-entities. We expand on this paper by validating a findings, noting discrepancies, and observing the model's behavior for inputs that were not provided by the paper's original authors.

1.4 Likely Challenges and Mitigations

One of the primary challenges in reproducing the study is handling the complexity of training generative models with specific tokenization and labeling practices, which may not align perfectly with readily available labeling tools. Another difficulty is the requirement for substantial computational resources, especially when implementing high-dimensional models like GNER-LLaMA and GNER-T5 for zero-shot learning across diverse

domains.

To mitigate these challenges, we plan to:

- Use tools like spaCy for some labeling tasks, although adjustments may be required to align with the paper's labeling format
- Run experiments with smaller datasets or quantized versions of models if full-scale experiments are computationally challenigng
- Document discrepancies in tokenization and labeling to ensure that any deviations from the original study are transparent for future replication efforts

2 Related Work

Existing works explore zero-shot performance, without incorporating non-entity text for negative instances, and focuses on various GPT models. This study implements LLaMA and Flan-T5 as base models.

A study noted zero-shot information extraction aims to build information extraction from text that is not annotated (Wei et al., 2024). They found models like GPT-3 and ChatGPT performed well when applying zero-shot methods. They applied a framework to ChatGPT where they transform zero-shot prompts to question-answering prompts for entity-relation triple extract, named entity recognition (NER), and event extraction. They found that performance increased from their applied framework. However, this study did not explore non-entity text and had limited NER data.

Recent works diversify their NER datasets, and investigate different applications for zero-shot NER. Firstly, UniversalNER used targeted distillation to train models using mission focused instruction tuning for a broad information extraction tasks (Zhou et al., 2024). Their dataset included biomedicine, programming, social media, law, and finance domains. They noticed improved performance and had a diverse dataset.

Secondly, GoLLIE was developed to solve the lack of generalization for unseen tasks applicable to zero-shot information extraction (Sainz et al., 2024). Their dataset included artifical intelligence (AI), literature, music, politics, and science domains. GoLLIE is able to better generalize by fine-tuning to comply with annotation guidelines. They found GoLLIE shows a smaller gap in F1 scores between the seen and unseen labels.

However, these works did not explore non-entity text. Non-entity text for negative instances is significant for classification models like BERT (Devlin et al., 2019). However, Rethinking Negative Instances for Generative Named Entity Recognition explores non-entity text for negative instances on generative models like LLaMA and Flan-T5, instead of classification models.

This study designed the framework Generative NER (GNER) to show that negative instances improve performance by incorporating context and making clearer distinctions between entities and non-entities. They attempt to handle text inaccuracies with converting unstructured text into structured entities. They experiment on Flan-T5 and LLaMA to create GNER-T5 and GNER-LLaMA, and found better performance than GoLLIE and UniversalNER.

3 Experiments

3.1 Datasets

The datasets used are available in the repository for this paper. No further data collection was performed.

3.2 Implementation

The repository for the re-implementation is `https://github.com/mamnuya/GNER.git`. This repository is forked from `https://github.com/yyDing1/GNER`.

3.3 Results

We attempt to implement these experiments using details from the paper. Firstly, This paper claimed that beam search enhances generative models. They include an example to demonstrate that the model corrects earlier mistakes when using beam search with a beam size of 2.

To validate figure 6, we implemented beam search with a beam size of 2 for the GNER-LLaMA model using Python. However, our results differ from the published results as shown in Table 1. We use spaCy module labels instead of the paper's labeling Bio-format prediction guidelines as this was not automatically produced from the model. The labeling guidelines were not specified in the context of a separate beam search implementation. We highlight the anticipated self-correction behavior, where additional tokens are generated to self-correct. Our labeling guidelines differ from those implemented by the paper. The

Table 1: Comparison of Figure 6

Test Case	Expected Output	Actual Output	Result
Final Prediction	What(O), was(O), the(B-title), fog(I-title), rated(O), ?(O)	What(O), was(O), the(O), fog(ORG), r(O), ated(O), ?(O), 2(CARDINAL), 0(CARDINAL), ...	Incorrect tokenization, excessive tokens produced
Ground Truth	What(O), was(O), the(B-title), fog(I-title), rated(O), ?(O)	What(O), was(O), the(O), fog(ORG), r(O), ated(O), ?(O), 2(CARDINAL), 0(CARDINAL), ..	Incorrect tokenization, excessive tokens produced
Highest Beam	What(O), was(O), the(O), fog(O)	What(O), was(O), the(O), fog(ORG)	Unable to incorporate label guidelines
Second Beam	What(O), was(O), the(B-title), fog(I-title)	What(O), was(O), the(O), fog(ORG)	Unable to incorporate label guidelines

paper includes Bio-format prediction guidelines for instruction tuning in prompts, but not for labeling existing strings that implement beam search separately.

The self-correction mechanism was expected to improve predictions by including additional relevant tokens through beam search. There is a discrepancy in word tokenization and labeling. Some tokens are split into subtokens by the tokenizer. Since we use spaCy for labeling, fog(ORG) occurs instead of fog(O) in our results. Additionally, many more tokens are produced than expected for the final predictions test and ground truth test.

However, excluding the excessive token productions, differing token labels, and splitting of tokens, the format of the actual output resembles the results in the paper. For instance, overlooking the several differences, the overall order and tokens produced in our beam search resembles the outputs from the paper. Our experiment produced nearly similar tokens in the same order. The labeling guidelines are not replicated as we implement spaCy for this beam search validation test.

To validate the paper's Table 10 Case 1 on GNER-LLaMA, our results differ from the published results as shown in Table 2. The tokenizer splits "directing" into "direct" and "ing." We explicitly code the the tokenization labeling in attempt to highlight the inconsistency in tokenization between the expect and actual outputs, which persists when applying beam search or not applying beach search. The tokenizer splits words unexpectedly which results in differing tokens. For example, with and without beam search, the token "directing" results in "direct" and "ing," which falters from the expected results, regardless of labels assigned.

To validate the paper's Table 9, we use 4-bit quantization on the models and write unit tests for each case of omission, addition, and substitution for the paper's 2 models. These tests do not incorporate any labeling guidelines since Table 9 does not use labeling guidelines outlined in the paper. There is not clear code nor instructions on how to implement the model to validate the respective test cases. However, in the attempt to load and generate output for Table 9 tests, the process results in 'Killed' on the GPU clusters of George Mason University's Office of Research Computing. This may indicate the computing resources are insufficient to support the tests for substitution, omission, and addition using the paper's models.

We edited the author's provided quick reproduction commands to reflect the correct model file paths. The commands print the F1-score, precision, and recall for various datasets of GNER-LLaMA and GNER-T5. These scores exactly match those of Table 2 in the published paper.

3.4 Discussion

Firstly, when attempting to download the necessary libraries, there was an error about modules. Furthermore, the README.md included preferred versions for modules. We corrected the README.md and input valid versions of the libraries into requirements.txt. Valid versions were defined as versions the original authors intended for use as specified in the README.md. We accordingly edited the README.md to include the command to download the needed libraries using requirements.txt. Furthermore, the README.md included commands for quick reproduction results. The file paths did not reflect the correct file paths, leading to a no such file or directory error. We corrected the file paths in the reproduction step of the README.md.

There were many inconsistencies to the paper results as we base our test and implementation on

Table 2: Comparison of Table 10 Case 1

Test Case	Expected Output	Actual Output	Result
With Beam Search	who(O), is(O), directing(O), the(B-title), hobbit(I-title)	who(O), is(O), direct(O), ing(O), the(B-title)	Incorrect tokenization
Without Beam Search	who(O), is(O), directing(O), the(O), hobbit(B-title)	who(O), is(O), direct(O), ing(O), the(B-title)	Incorrect tokenization

details provided in the paper. We did not need to implement the labeling guidelines outlined in the paper to verify table 9 as the labeling guidelines is incorporated into instruction tuning and generation prompts, and we encoded strings without the instruction tuning prompt. For the table 9 validation tasks, the process resulted in 'Killed' despite multiple attempts to reduce the amount of memory and compute utilized, and accessing different GPU configurations.

For labeling tasks when implementing beam search, the labeling technique was unclear. We used spaCy for labeling instead of generating labels using the model's generation prompts since we implement beam search separately.

3.5 Resources

We utilized GPU resources from George Mason University's Office of Research Computing. These resources were sufficient for various testing tasks on the model, however, insufficient for some generations from the models of the published paper. This project took significant development effort. The experiments ran for a long time even on quantized versions of the model, on GPU and on CPU.

4 Error Analysis of Generative Named Entity Recognition Models

Named Entity Recognition (NER) is a crucial subtask in natural language processing (NLP) that focuses on identifying and classifying entities such as names of people, organizations, locations, and other predefined categories in text. Generative NER models like GNER-LLaMA and GNER-T5 aim to enhance this task by leveraging generative approaches to predict labels for tokens in input sequences. These models operate in diverse contexts, where synonym recognition and context-aware entity identification are critical for accurate classification.

This study conducts an error analysis of the GNER-LLaMA-7B model in handling synonym-based entity recognition. The objective is to evaluate its performance in scenarios where sentences contain synonyms or semantically related terms that correspond to specific entity labels. Such analysis provides insights into the model's strengths and limitations, particularly in generalization, vocabulary coverage, and token-label alignment.

The analysis involves running a set of test sentences through the GNER-LLaMA-7B model and comparing the predicted entity labels against expected labels. The results reveal areas where the model struggles, offering recommendations to improve its performance through fine-tuning, expanded vocabulary, and better token alignment strategies.

This section provides an in-depth analysis of the errors encountered during the evaluation of Generative Named Entity Recognition (NER) models, specifically GNER-LLaMA and related configurations. The focus is on tokenization, resource usage, contextual entity recognition, synonym recognition, and labeling accuracy. Each subsection elaborates on observed patterns, specific examples, and recommendations.

4.1 Error Analysis of Tokenization Test

Tokenization plays a crucial role in NER models, as errors in token splitting propagate to downstream tasks. Our evaluation revealed several challenges:

1. **Ambiguity in token splitting:** The tokenizer showed significant difficulty in segmenting complex words into appropriate subwords. This error affects models trained on tokenized inputs.
   Example: For the input "Analyzing multi-token words like re-directing," the tokenizer split the word into '['Anal', 'yz', 'ing', 'multi', '-', 'token', 'words', 'like', 're', '-', 'direct', 'ing']', which deviated from the expected output.

2. **Hyphenated phrases:** Words such as "high-profile" and "end-to-end" were inconsistently segmented, leading to mismatched tokens

that affect context interpretation.
Example: The phrase "end-to-end encryption" resulted in tokens '['Ident', 'ifying', 'end', '-', 'to', '-', 'end', 'encryption', 'methods']', diverging from expected outputs.

3. **Impact on subword embeddings:** Incorrect token splits affect the embeddings generated for subwords, which hinders the model's ability to accurately identify entities within a sentence.

4.2 Error Analysis of Resource Usage Test

Efficient resource utilization is critical for deploying NER models in real-world applications. The resource usage test revealed:

1. **High memory overhead:** The GNER-LLaMA model exhibited excessive memory usage, peaking at over 27 GB during evaluation. This indicates potential inefficiencies in handling large parameter spaces.
Example: Memory usage consistently ranged between 27,419 MB and 27,451 MB over multiple iterations.

2. **Static resource consumption:** Despite varying input complexities, the memory usage showed little dynamic adjustment, suggesting limited architectural optimization for resource-constrained environments.

3. **Impact on scalability:** These results indicate challenges in deploying such models on devices with limited computational resources, such as edge devices or mobile platforms.

4.3 Error Analysis of Contextual Entity Recognition Test

Contextual understanding is a fundamental requirement for effective NER. The results highlighted:

1. **Contextual ambiguity:** The model failed to assign appropriate labels for contextually ambiguous sentences.
Example: For the sentence "George Washington was the first president of the United States," all tokens were labeled as 'LABEL_0', indicating no entity recognition.

2. **Lack of contextual differentiation:** The model struggled with sentences where entities depended heavily on context.
Example: In "I am flying to Washington for the conference next week," "Washington" was expected to be labeled as 'B-LOC' but was misclassified.

3. **Misalignment with pretraining:** The mismatch between model pretraining objectives and the downstream task was evident in its failure to distinguish between person and location entities effectively.

4.4 Error Analysis of Synonym Recognition Test

Synonym recognition is critical for ensuring robust generalization. However, the analysis revealed:

1. **Vocabulary mismatches:** The model's inability to generalize across synonymous terms led to significant errors.
Example: The phrase "The Prime Minister of the UK made a speech" failed to map "Prime Minister" to the expected label 'B-TITLE'.

2. **Contextual synonym recognition:** The model demonstrated limited ability to handle synonyms in context-sensitive scenarios, such as distinguishing between synonyms of geographical locations.

3. **Pretraining data limitations:** The synonym errors highlight potential inadequacies in the diversity of the pretraining dataset.

4.5 Error Analysis of Labeling Test

Labeling accuracy is critical for evaluating the overall performance of NER models. The analysis found:

1. **Accuracy for straightforward cases:** The spaCy model performed exceptionally well for simple sentences.
Example: For the sentence "Did George Clooney make a musical in the 1980s?", the predicted labels perfectly matched the expected labels.

2. **Potential weaknesses in complex scenarios:** While the current test cases did not include overlapping or nested entities, the model's performance for such cases remains unexplored.

5 Robustness Study

To evaluate robustness, we apply the evaluation script provided by the authors and examine behavior when we provide .input files with no predictions, predictions with missing fields, predictions with limited data and minor word edits, and observe the behavior.

For robustness, we test the model on English, Spanish, and Bangla strings, as well as strings of special characters. We implement these tests using snippets of author provided code in their repository.

We provide a prompt in English provided by the author's as an example in their README.md as model input. Then we write tests to verify the model outputs anticipated labels as outlined in the README.md and paper. If labels are not clearly defined, we assume they will be labeled with (O). We check that the model outputs labels for the inputs without failures or system errors.

We adapt the English test to instead prompt in Spanish and write tests based on the logic of the expected English labels.

We also translate the prompt into Bengali and write tests expecting the model to not recognize the characters nor tokens since the model is not tested for different language characters nor tokens. Therefore, we expect the output to result in labels of (O) indicating the word does not belong within the specified entity labels.

Additionally, we create a prompt of strictly special characters and write tests expecting the model to not recognize these characters nor tokens. Therefore, we expect the output to result in labels of (O) for every space-separated string of special characters.

5.1 Results of Robustness Evaluation

When inputting a file that is missing the dataset field, we observe a KeyError. When inputting a file that is missing all fields, we observe a ZeroDivisionError. When inputting a file with limited datasets and editing one word from their reproduction results, we found the evaluation script works for limited data.

We determine robustness by testing if the model provides the anticipated Bio-format predictions for English, Spanish, Bangla, and special character inputs. The results are illustrated in 3.

For English and Spanish, the model nearly performs as expected. The words are predicted with the expected Bio-format labels. However for the English test, the model considers the last "?" to be a separate token and labels this with (O) in our tests. The README.md from the author's example shows this exact test returning with no label for the last question mark.

For the Spanish test, the model does not label the beginning "¿" as a separate token, however, the model considers the last "?" to be a separate token and labels this with (O) in our tests. Based on the logic of the English example from the author's README.md, our test expect no label for the last question mark.

For the Bangla test, we expected the model to label all words with (O) since the model is not expected to recognize Bangla words with entity labels. However, the model recognizes the Bangla representation for the year. Additionally, it marks words that would be marked with "B-Actor, I-Actor" with "B-character, I-character." This behavior was undefined. The rest of the tokens were labeled with (O). Additionally, the model considers the last "?" to be a separate token and labels this with (O) in our tests. Based on the logic of the English example from the author's README.md, our test expect no label for the last "?"

For the special characters test, there are 5 space separated strings of special characters. The tests expected that the 5 substrings would be represented with 5 (O) labels. However, the model labeled each of the 13 special characters with (O).

The model does not crash for these prompts. The model fails to recognize and label entities correctly in Bangla script, and succeeds for Spanish and English. The model always succeeds in recognizing numerals for the year entity label. The paper did not clearly define the behavior for special characters causing a discrepancy in labeling special characters. The model labeled every special character with (O) in our robustness tests. There are examples in the README.md with some special characters labeled and some special characters not labeled.

6 Discussion

6.1 Robustness Testing

For the robustness testing, the special characters test labeling each special character with (O) was unexpected based on the author's provided demo usage. Additionally, recognition of Bangla numbers was unexpected. There is an actor in Bangla

Table 3: Robustness Test Results for English, Spanish, Bangla, and Special Characters

Test Case	Input	Expected Output	Actual Output	Result
English	Did George Clooney make a musical in the 1980s?	Did(O), George(B-actor), Clooney(I-actor), make(O), a(O), musical(B-genre), in(O), the(O), 1980s(B-year)	Did(O), George(B-actor), Clooney(I-actor), make(O), a(O), musical(B-genre), in(O), the(O), 1980s(B-year), ?(O)	Unexpected label for "?"
Spanish	¿George Clooney hizo un musical en los años 1980?	George(B-actor), Clooney(I-actor), hizo(O), un(O), musical(B-genre), en(O), los(O), años(O), 1980(B-year)	George(B-actor), Clooney(I-actor), hizo(O), un(O), musical(B-genre), en(O), los(O), años(O), 1980(B-year), ?(O)	Unexpected label for "?"
Bangla	জর্জ ক্লুনি কি ১৯৮০ সালের দশকে কোনো মিউজিকাল বানিয়েছিলেন ?	জর্জ(O) ক্লুনি(O) কি(O)১৯৮০(O) সালের(O) দশকে(O) কোনো(O) মিউজিকাল(O) বানিয়েছিলেন(O)	জর্জ(B-character) ক্লুনি(I-character) কি(O) ১৯৮০(B-year) সালের(O) দশকে(O) কোনো(O) মিউজিকাল(O) বানিয়েছিলেন(O) ?(O)	Unexpected label for "?" and recognition of Bangla numerals, incorrect labels for actor
Special Characters	!*! @^& *(*($%$	!*!(O) @^&(O) *((O) *((O) $%$(O)	!(O) *(O) !(O) @(O) ^(O) &(O) *(O) ((O) *(O) ((O) $(O) %(O) $(O)	Unexpected label for each special character

script that is labeled with character entity labels, which should have received an actor entity label or an (O) label. Additionally, the ending question mark for each input regardless of language is labeled with (O) which is not reflective of the behavior outlined in the paper and some examples in the README.md. However, other examples in the README.md show this behavior. This shows an unexplained inconsistency with special characters.

6.2 Experiments

For the experiments, we would benefit from more compute resources. If we had more time on this project, we would like to have secured more powerful compute resources, decipher how to implement the model to test table 9 results, and run tests for table 9 of this paper. However, the tests, memory, or compute restraints prevented the verification of table 9. Furthermore, we based most experiments on implementation details mentioned in the paper, which lacked critical information for effectively reproducing results found in the paper. However, once we corrected the file paths, the author provided commands for quick reproduction reflected the F1-score, precision, and recall scores from the paper.

6.3 Recommendations for Code and Description Clarity

For future developers, the README.md and paper should outline and reflect the behavior of the code. If there might be discrepancies by user input, they should be clearly defined. All file paths and commands should be updated based on the current state of the code repository. Furthermore, editable code or detailed descriptions should be provided for findings, which would have helped for implementing the expected results for beam search and other experiments.

6.4 Recommendations for Model Improvement

To address the identified challenges, we propose the following:

- **Fine-tuning with domain-specific datasets:** Enhance the model's robustness by training it on datasets that include diverse, ambiguous, and multilingual entities.
- **Tokenizer improvements:** Redesign the tokenizer to handle complex word structures, such as hyphenated and multi-token phrases, with higher accuracy.
- **Efficient architectures:** Introduce parameter-efficient transformer variants

to optimize memory usage and computational overhead.

- **Robust evaluation framework:** Develop an evaluation framework that includes challenging test cases with nested, overlapping, and multilingual entities.
- **Augment synonym recognition capabilities:** Train models with datasets that explicitly include synonymous entities across various domains and contexts.

6.5 Recommendations for Future Work

To build on the current findings, future research could focus on:

- Implementing cross-lingual NER to evaluate performance on underrepresented languages.
- Extending the evaluation to include nested entities and overlapping annotations.
- Exploring lightweight alternatives to GNER-LLaMA or GNER-T5 for deployment on edge devices.
- Conducting ablation studies to identify the most influential factors affecting model performance.

7 Workload Clarification

Describe how the team divides the workload in this checkpoint. Note that each team member should contribute roughly the same amount of work to this assignment.

Mamnuya Rinki completed the following tasks:

- Shortlisted papers from ACL Anthology 2024
- Created the GitHub for collaboration
- Completed Sections: 2 Related Works, 3 Experiments, 5 Robustness Study, 7 Workload Clarification, and 8 Conclusion
- Completed Subsections: 6.1, 6.2, 6.3
- Created tables for report
- Created requirements.txt
- Incorporated requirements.txt, corrected file paths, and reproduction commands into README.md
- Performed implementation, testing, analysis, and writeup for verifying the paper's figure 6, case 1 of table 10, table 9, and table 2.
- Performed implementation, testing, analysis, and writeup for robustness testing, including tests for English, Spanish, Bangla, and special characters
- Performed implementation, testing, analysis, and writeup for author provided evaluation script on inputs with no predictions, predictions with missing fields, and predictions with limited data and with word edits
- Reviewed error analysis code using BERT models, updated with GNER-LLaMA model and tokenizer, and changed error analysis section in README.md to reflect GNER-LLaMA model and tokenizer usage

Mary Rithika Reddy Gade completed the following tasks:

- Completed Introduction Section 1
- Completed Subsections: 6.4, 6.5
- Updated README.md with error analysis section and findings
- Edited requirements.txt to support error analysis libraries
- Performed implementation, testing, analysis, and writeup for error analysis in Section 4 "Error Analysis of Generative Named Entity Recognition Models," including Contextual Entity Recognition, Synonym Entity, Tokenization, Resource Constraint, and Labeling Tests
- Provided feedback on table placement in report

8 Conclusion

The paper is somewhat reproducible. Implementing code and tests on our own based on details in the paper resulted in many complications for the experiments. For robustness testing, there were less complications when using snippets of author provided code in their repository. The paper itself is not sufficient to replicate results due to critical details overlooked from the code.

For English inputs to the model, the results align with the labels assigned using the prompt provided by the author's. There are assigned labels for special characters which was not anticipated based on the author's provided demo usage, but the overall labels in these cases are consistent with the paper. The F1-scores, precision, and recall metrics were reproducible using the author's provided commands after we edited the command to reflect the correct file paths.

The paper shows beam search aids in cases of substitution, omission, and addition. The beam search results were difficult to reproduce by implementing on our own due to lack of detailed information. When attempting to validate results by implementing beam search separately, we encountered differences in tokenization, unclear methods to label strings when implementing beam search separately, and in some cases experienced too many tokens being produced. However, overlooking some of these differences, our beam search tests produced subtokens in the same order as the results in the paper. The overall order and tokens produced in our beam search resembles the outputs from the paper, excluding inconsistencies in labeling, tokenization, and number of tokens produced. Some tests were not suitable for our GPUs nor CPUs, resulting in inconclusive results, such as those for the paper's Table 9.

Error testing demonstrated that tokenization did not align with expectations. Furthermore, the model failed to assign appropriate labels for contextually ambiguous sentences and to recognize synonyms. There is potential to explore overlapping or nested entities.

Robustness testing revealed problems with multilingual inputs. Additionally, special characters were given labels although this was not consistent with outputs within the paper and code samples.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Yuyang Ding, Juntao Li, Pinzheng Wang, Zecheng Tang, Bowen Yan, and Min Zhang. 2024. Rethinking negative instances for generative named entity recognition.

Oscar Sainz, Iker García-Ferrero, Rodrigo Agerri, Oier Lopez de Lacalle, German Rigau, and Eneko Agirre. 2024. Gollie: Annotation guidelines improve zero-shot information-extraction.

Xiang Wei, Xingyu Cui, Ning Cheng, Xiaobin Wang, Xin Zhang, Shen Huang, Pengjun Xie, Jinan Xu, Yufeng Chen, Meishan Zhang, Yong Jiang, and Wenjuan Han. 2024. Chatie: Zero-shot information extraction via chatting with chatgpt.

Wenxuan Zhou, Sheng Zhang, Yu Gu, Muhao Chen, and Hoifung Poon. 2024. Universalner: Targeted distillation from large language models for open named entity recognition.