Adversarial Named-Entity Recognition with Word Attributions and Disentanglement

Xiaomeng Jin
UIUC
xjin17@illinois.edu

Bhanukiran Vinzamuri
Amazon
vinzamub@amazon.com

Sriram Venkatapathy
Amazon
vesriram@amazon.com

Heng Ji
Amazon
jihj@amazon.com

Pradeep Natarajan
Amazon
natarap@amazon.com

Abstract

The issue of enhancing the robustness of Named Entity Recognition (NER) models against adversarial attacks has recently gained significant attention (Simoncini and Spanakis, 2021; Lin et al., 2021). The existing techniques for robustifying NER models rely on exhaustive perturbation of the input training data to generate adversarial examples, often resulting in adversarial examples that are not semantically equivalent to the original. In this paper, we employ word attributions guided perturbations that generate adversarial examples with a comparable attack rates but at a lower modification rate. Our approach also uses disentanglement of entity and non-entity word representations as a mechanism to generate diverse and unbiased adversarial examples. Adversarial training results based on our method improves the F1 score over originally trained NER model by **8**% and **18**% on *CoNLL-2003* and *Ontonotes 5.0* datasets respectively.

1 Introduction

Named Entity Recognition (NER) seeks to identify and classify named entities mentioned in unstructured text into pre-defined categories such as *Person*, *Location*, or *Organization*. Such models can be brittle to minor perturbations to the input. Recently, there has been interest in developing adversarial attack-based techniques for NER models (Simoncini and Spanakis, 2021; Lin et al., 2021) to robustify these models. The existing word-level attack methods perturb words across the entire sentence to generate adversarial examples. While a small fraction of such perturbations do result in a successful attack, many of the perturbed sentences are either less effective or do not qualify to be adversarial examples i.e., they are not semantically similar to original sentences. Table 1 shows an example of a range of potential adversarial examples can be generated through exhaustive perturbations.

Original	Ronaldo/PER will play nine matches against Manchester/ORG United/ORG.
Ineffective	Ronaldo/PER will play **ten** matches against Manchester/ORG United/ORG.
Improper	Ronaldo/ORG will play nine matches **for** Manchester/ORG United/ORG.
Good	Ronaldo/ORG will play nine **games** against Manchester/ORG United/ORG.
Good	**Messi**/ORG will play nine matches against Manchester/ORG United/ORG.

Table 1: Adversarial examples corresponding to a given input sentence.

We propose an approach for word-level adversarial attacks that generates adversarial examples in a principle manner. The perturbations focus on those words in the sentence whose absence can cause the label to change. In the input example presented in Table 1, the most important words influencing the NER in the sentence are 'Ronaldo', 'Manchester', 'United' and 'matches' and hence, are replaced by relevant substitutes to generate the adversarial examples. The disentanglement of the latent representations of entity or non-entity word types prevents bias towards perturbations of a particular word type. The guided generation of the adversarial examples also makes them more explainable. For example, the label flip as a result of replacing 'matches' with 'games' in the above example indicates over-reliance on the word 'matches' for tagging entities.

The workflow illustrating our approach is presented in Figure 1. The approach consists of the following steps, (1) Disentanglement of word representations of entity and non-entity words by minimizing difference in reconstruction errors in an auto-encoder setup, (2) Selection of words to permute based on their word attribution scores computed with gradient-based techniques such as Integrated Gradients (Sundararajan et al., 2017) used

in this paper, (3) Substitution of the selected words with candidate alternatives based on their word and entity types, (4) Selection of candidate adversarial examples based on their semantic similarity scores to the original text, and (5) Adversarial training using the selected adversarial examples for improved NER model robustness. We extensively evaluate the approach presented in this paper on two popular datasets (*CoNLL-2003* (Sang and De Meulder, 2003) and *Ontonotes 5.0* (Weischedel et al., 2013)). The empirical results on the downstream NER task shows that our method improves the F1 score over originally trained NER model by **8**% and **18**% on *CoNLL-2003* and *Ontonotes 5.0* respectively. On intrinsic evaluation of our adversarial examples generation approach on *CoNLL-2003*, we achieve **10%** higher attack success rate (percentage of generated adversarial examples that cause label flip) at a comparable modification rate.

The main contributions in this paper are,

- Our approach of generating adversarial examples with word attributions and disentanglement achieves comparable attack success rates but at an overall lower modification rate compared to the SOTA baselines.
- We introduce an approach to learn disentangled representation of entity and non-entity words, and utilize it to generate a set of diverse adversarial examples. Our results show that we achieve a more balanced choice of entity and non-entity words for perturbation.
- We study different attack strategies focusing on perturbing different word types: entities, non-entities or both. The choice of the attack strategy can depend on the modification rate that we would like to achieve.

The rest of the paper is structured as follows. In Section 2, we survey prior art in this space and highlight how our contribution differs. In Section 3, we present our approach and discuss the individual technical components on disentanglement, influence functions, semantic similarity and adversarial training. In Section 4, we present attack and adversarial training results on two benchmark datasets followed by concluding in Section 5.

2 Related Work

The existing adversarial attack methods on NER tasks can be classified into three categories: Character-level Attack, Word-level Attack, and Sentence-level Attack. The Character-level attacks generate adversarial examples by adding, deleting, or replacing a character in a word in the natural language texts. HotFlip (Ebrahimi et al., 2017) performs a white-box character-level attack by swapping characters based on their gradient with respect to a one-hot input representation. DeepWordBug (Gao et al., 2018) proposes a black-box attack by designing token scoring strategies and replacing tokens that minimize the edit distance of the perturbation. However, these attacks have the problem that the character swapping generates spelling typos and the sentences are not semantically meaningful. Sentence-level attacks perform the operations by altering the input texts on the whole sentence. SCPN (Iyyer et al., 2018) includes a paraphrase generation under the guidance of a trained parser to label the syntactic transformation. (Lei et al., 2019) proposes a paraphrasing greedy algorithm using the gradient of the attacked classifier. However, sentence-level attacks generally require a larger-scaled model and the generated texts are also more distant to the original inputs.

Word-level attacks are more popular and effective methods than the above-mentioned two methods. There are also many effective word-level attacks on text classification tasks (Ribeiro et al., 2020; Das and Paik, 2022). (Zang et al., 2019) designs a word-level attack that incorporates a sememe-based word substitution method and a swarm optimization-based search algorithm. (Liu et al., 2021) proposes an efficient local search algorithm to determine the possible word substitutions. However, the adversarial attacks on sequence-to-sequence models are not much explored. (Simoncini and Spanakis, 2021) extends TextAttack (Morris et al., 2020) framework that consists of multiple attack strategies via reformulating the goal functions to support NER tasks. RockNER (Lin et al., 2021) is a simple NER adversarial attack by perturbing both named entities and contexts in the original texts. However, the generated adversarial examples all have the issue of poor semantic equivalence to the original input sentences because of the high modification rate. In contrast, we propose a word-level adversarial attack on NER models that effectively identifies important words via word attributions and generates the adversarial examples with a lower modification rate.

Disentanglement (Higgins et al., 2017) has been primarily used in the space of auto-encoders for learning latent factors which are mutually orthog-

onal (diverse) w.r.t one another to aid with interpretability primarily for image-based applications. They have also been extended for text-based tasks (Zou et al., 2022) to leverage the latent components to learn more desired downstream representations. In this paper, we use disentanglement to segregate the entity and non-entity representations in the feature space.

3 Our Method

In this section, we describe key technical components of our proposed adversarial NER framework. For the sake of brevity, we refer to *non-entity* as *context* throughout the remainder of this paper. The approach consists of the following steps, (1) Disentanglement of word representations, (2) Selection of important words to permute, (3) Substitution of the selected words with candidate alternatives, (4) Selection of candidate adversarial examples, and (5) Adversarial training.

3.1 Disentanglement of Word Representations

The primary purpose of disentanglement is to reduce the bias (between entity and context words) in the selection of words for perturbation, thereby increasing the diversity of the generated adversarial examples. In Table 2, we notice that the NER models are highly biased on context words when predicting the NER labels. The word selection approach (described in detail in Section 3.3) considers mostly (98.9% for CoNLL 2003 dataset in Table 2) the contextual words for generating potential adversarial examples if disentanglement is not applied. This adversely affects the overall diversity of the generated adversarial examples thereby preventing exploration of potential vulnerabilities of the NER model to entity specific perturbations. The approach for disentanglement mitigates this problem by enabling a greater balance in the selection of context and entities for generating adversarial examples. It works by segregating the context and entity words in the embedding space. Figure 1 depicts the impact of disentanglement on the word representations as visualized through Uniform Manifold Approximation (UMAP) (Leland McInnes, 2020). We see that the word representations of the entity words after disentanglement are clearly separable from the context words.

We now outline our approach for learning the disentangled representations before computing the word attributions. Given the original features of an input sentence, we first split word features into two sets E and C, where E represents features of entity words and C represents features of context words. Our goal is to learn a new representation $\hat{E}$ for entity words, so that the features of entity words are independent of context words. The representation $(\hat{E}, C)$ is used for computing word attribution scores in the next step.

Dataset	CoNLL2003	OntoNotes
Original Distribution of Context	83.2%	91.1%
Distribution of Context as most important words before disentanglement	98.9%	96.1%
Distribution of Context as most important words after disentanglement	81.3%	91.4%

Table 2: The first row represents the original distribution of with the NER label O (context words) among all word labels in the two datasets followed by two rows representing the distribution of important context words identified by the IG word attribution function in two different settings. It is clear that in comparison to before disentanglement (the second row), the representation learned using our balancing disentanglement techinique (in third row) obtains a distribution that aligns more closely with the original dataset distribution (first row).

Following (Marx et al., 2019), we learn the disentangled representations by using an auto-encoder. The architecture of the auto-encoder is illustrated in Figure 2, which consists of three neural networks: the encoder, the decoder, and the discriminator:

Encoder. The encoder Enc takes entity word embeddings E as input, then learn a disentangled representation E' in the latent space. Note that the encoder is also aware of context word embeddings C when encoding E:

$$Enc(E; C) = E'. \tag{1}$$

Decoder. The decoder Dec takes E' as input and output $\hat{E}$, which is a new (disentangled) representation for E:

$$Dec(E'; C) = \hat{E}. \tag{2}$$

Discriminator. The discriminator Dis tries to predict (recover) the context embeddings C using the hidden representation of entity words E':

$$Dis(E') = \hat{C}, \tag{3}$$

where $\hat{C}$ is the predicted result for C. However, since we expect E' and C to be independent, the

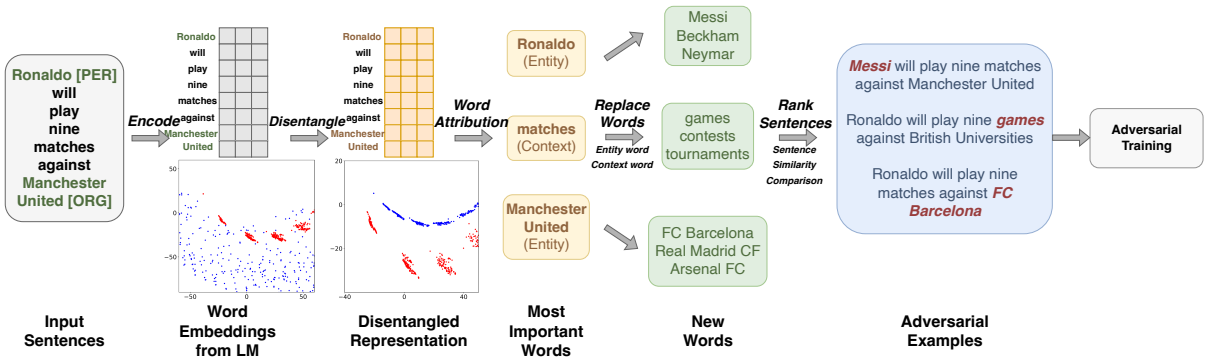

Figure 1: The architecture of our method as depicted here proceeds as follows. Given an input text, we first obtain word embeddings from language models that the NER model uses, then learn a disentangled representation for entity and context representations (context, entity in the UMAP embedding scatter plots). The disentangled representations are then used to compute scores via the IG word attribution function. The most important words are replaced with new words following word substitution workflows. After generating new sentences, we rank the sentence similarity scores and output the most similar sentences as adversarial examples. Our framework outputs the adversarial examples for adversarial training to improve the NER models' robustness.

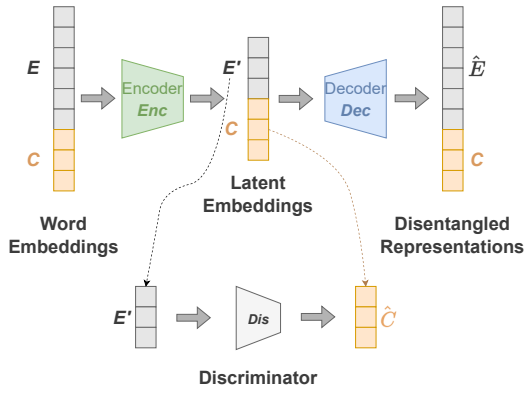

Figure 2: The architecture of disentangled representation learning framework which takes initial word feature embeddings as inputs, and use an auto-encoder to learn a latent representation ***E'*** for entity words. A discriminator *Dis* is used to recover context embeddings ***C*** from entity embeddings ***E'***

.

recovery error between $\hat{C}$ and C should be maximized. In other words, the discriminator cannot recover any information of C using E', which indicates that there is no correlation between C and E' (as well as $\hat{E}$).

Training objective. Our training process aims to minimize the reconstruction mean squared error (MSE) between E and $\hat{E}$, as well as maximizing the recovery error between C and $\hat{C}$:

$$L = \text{MSE}(E, \hat{E}) - \beta \cdot \text{MSE}(C, \hat{C}), \quad (4)$$

where β is a balancing hyper-parameter. $\hat{E}$ and C are taken as the disentangled representations for entity and context words.

3.2 Selection of Important Words

To evaluate how much the feature of each word affects the result of NER models, we propose to compute the word attribution scores (a.k.a feature importances or saliency scores) using Integrated Gradients (IG) (Sundararajan et al., 2017). It computes a gradient of the model's prediction output to its input features and returns the attributions of output labels with respect to the input features. The assigned value on each input feature is the importance score to model outputs. The mathematical formulation for IG is as follows

Suppose we have a black-box machine learning model f, an input $x \in R^n$, and a baseline input $x' \in R^n$ (for text models it could be a zero embedding vector). The Integrated Gradient (IG) along the i^{th} dimension for the input x and x' is defined as:

$$IG_i(x) ::= (x_i - x'_i) \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha(x - x'))}{\partial x_i}. \quad (5)$$

We obtain a ranked list of words in the sentence based on their word attribution scores. The top-K among them are then selected for perturbations. The use of disentangled word representations (as described in the earlier section) enables a more balanced selection.

3.3 Word Substitution of Selected Words

After identifying the most important words in a sentence, the next step is to determine possible sub-

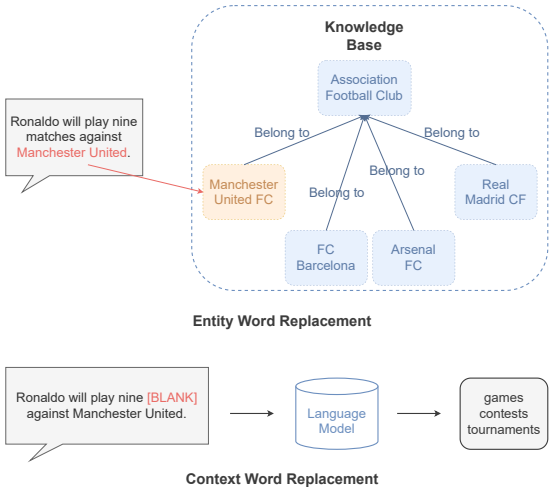

Figure 3: Entity and Context Word Substitution Workflows

stitutions for them. We outline both replacement workflows in Figure 3 and briefly describe them below.

Named Entity Substitution To keep the entity labels unchanged, the new named entity word should belong to the same entity type as the original entity word. Since Wikidata contains abundant structured knowledge and most of the entities have corresponding entries in it, we use it as as the external knowledge base for word replacement. Specifically, similar to (Lin et al., 2021), to determine the substitution for a given named entity, we first use entity linking tools (Honnibal et al., 2020) to link the named entity to an entry in Wikidata. If there is a matching entry in Wikidata, we collect the entries that have a *belonged to* relation with this entry and take them as the upper categories of the target named entity. We randomly sample at most 10 of the entries under the same upper categories as possible word substitutions.

As an example illustrated in Figure 3, the label of the target word **Manchester United** is an organization. We link this entity to the *Manchester United FC* entry in the Wikidata knowledge base. *Manchester United FC* belongs to an *Association Football Club* and under the same category of *Association Football Club* there are other entries such as *FC Barcelona*, *Real Madrid CF*, and *Arsenal FC*. These are possible substitutions to replace the entity word **Manchester United**.

Context Word Substitution We leverage an infilling language modeling framework (ILM) (Donahue et al., 2020) that allows language models fill in the blanks given masked texts. We use a fine-tuned language model GPT-2 (Radford et al., 2019) to generate texts. Specifically, given a context word to be replaced, we mask this word with the special token [BLANK] and pass the sentence into the ILM framework.The model then performs infilling, and we collect the top 5 new infilled words ranked by the language model as possible substitutions. as illustrated in Figure 3.

3.4 Sentence Ranking

For an input sentence, we identify target words to be substituted and replace them with new words, which gives us a set of new sentences. To decide which sentences should be output as the final adversarial samples, we compute the similarity between each new sentence and the original one using **Universal Sentence Encoder** (Cer et al., 2018). Universal Sentence Encoder (USE) encodes natural language texts into high dimensional vectors as text embedding. After encoding each pair of original and generated sentences, we compute their cosine similarity as their similarity score. Basically, given the original and new sentences S_1 and S_2, the similarity score is computed by:

$$Sim(S_1, S_2) = \frac{USE(S_1) \cdot USE(S_2)}{\|USE(S_1)\| \, \|USE(S_2)\|} \quad (6)$$

4 Experiments

4.1 Datasets

We conduct our experiments on two datasets, *CoNLL-2003* (Sang and De Meulder, 2003) and *Ontonotes 5.0* (Weischedel et al., 2013). *CoNLL-2003* is a named entity recognition dataset that has four types of named entities: persons, locations, organizations, and names of miscellaneous entities that do not belong to the previous three groups. *Ontonotes 5.0* is a large corpus containing news articles in three languages and its annotation has 18 types of named entities.

4.2 Baseline Methods

We compare our proposed method with the following NER adversarial attack baseline methods:

- **RockNER** (Lin et al., 2021) creates adversarial examples by operating at the entity level and replacing all target entities with other entities of the same semantic class in Wikidata. At the context level, it randomly masks up to

Attack Name	F1 Score ↓	Attack Rate ↑	Modification Rate ↓	Textual Similarity ↑
Bert-Attack	79%	44%	22%	84%
CLARE	79%	37%	70%	86%
DeepWordBug-II$_5$	89%	27%	18%	86%
DeepWordBug-II$_{30}$	87%	30%	21%	83%
SCPN	91%	18%	66%	59%
Our Method	82%	36%	11% (max 1 word)	91%
	81%	45%	13% (max 2 words)	90%
	79%	**54%**	22% (max 3 words)	84%
Original	98%	-	-	-

Table 3: Results under different attack strategies on *CoNLL-2003* dataset. The first 5 attack methods come from SeqAttack. We generate adversarial examples under different levels of modification rate where we allow at most 1 word, 2 words, or 3 words replacement. *Lower* F1 score and modification rate along with *higher* attack rate and textual similarity indicate a more superior attack strategy as demonstrated by our method.

Model Name	F1 Score	F1 Score		
	Original test	Context-only ↓	Entity-only ↓	Context + Entity ↓
BERT-CRF (RockNER)	90.6%	85.8%	59.2%	54.6%
BERT-CRF (Ours)	90.3%	79.6%	51.1%	47.5%

Table 4: F1 score under attack results on *OntoNotes* dataset. Both RockNER and our method are evaluated under context-only attack, entity only attack, and context+entity attack settings. Comparing numbers row-wise against RockNER (first row) reveals that our method (second row) obtains *lower* F1 score compared to F1 score on original test set under all three attack settings.

Dataset	CoNLL2003	OntoNotes
Train	14,987	59,924
Validation	3,466	8,528
Test	3,684	8,262

Table 5: Train, Validation and Test splits on *CoNLL-2003* and *Ontonotes 5.0* datasets.

3 context words and uses pre-trained language models (PLM) to generate word substitutions.

- **SeqAttack** (Simoncini and Spanakis, 2021) is an attack framework against token classification models. The framework extends TextAttack (Morris et al., 2020) and contains multiple adversarial attack strategies. We compare our attack method with 5 attacks in the framework, including Bert-Attack, CLARE, DeepWordBug, and SCPN.

4.3 Experimental Setup

We evaluate the effectiveness of our attack methods based on multiple evaluation metrics. Finally, we present adversarial training results to indicate the the improvement in model robustness using our method.

4.3.1 Evaluation Metrics

- **Attack success rate**. An attack is successful on a sentence if there is at least one named entity in the sentence which is misclassified by the NER model after attack. We calculate the rate of successful attacks among all attacks.
- **Modification Rate**. For an original sentence and the modified sentence after attack, we calculate the percentage of different tokens between the two sentences as the modification rate.
- **Textual Similarity**. We compute the textual similarity between the original sentence and the generated sentence. We use USE to encode each sentence and compute cosine similarity between the vectors.

4.3.2 Attack and Hyper-parameter Settings

For our method, we conduct context-only attacks that allow perturbing context words only, entity-only attacks replacing entity words only, as well as context + entity attacks allowing replacement across both. In Table 3, to test the attack performance of our method, we report results at different levels of modification rate (maximum 1 word, 2 words or 3 words). All displayed results are aver-

Original Test Sentence	Adv Example (RockNER)	Adv Example (Our Method)
Dear viewers, the China News **[ORG]** program will end here.	Dear viewers, the *Hiwwe wie Driwwe* **[O]** *people* will *lose* here.	Dear viewers, *my* China News **[WORK-OF-ART]** program will end here.
Relevant departments from Beijing Municipality **[GPE]** promptly activated emergency contingency plans.	*Related* departments from *Markham* **[O]** promptly activated emergency contingency *members*.	Relevant departments from *Berlin* Municipality **[ORG]** promptly activated emergency contingency plans.

Table 6: A case study on generated adversarial examples by our method. The sentences on the left are original test sentences from *OntoNotes* dataset with correctly predicted named entities (blue colored words). The sentences in the middle and on the right are generated adversarial examples using RockNER baseline and our method that cause the NER model make wrong predictions. The *italic* words are the candidate words got selected and replaced. The red labels are incorrect predictions after the adversarial attacks ([O] means *Not an Entity*).

aged across 3 random runs.

The encoder, decoder and discriminator for computing the disentangled representations consist of two hidden layers with size = [64, 10] and β was set to 0.5. For IG word importance computation, we leverage the Captum library from Meta (Kokhlikyan et al., 2020). The learning rate is 1e-3, and the number of epochs is 25. We use Stochastic Gradient Descent optimizer with weight decay 1e-5.

After computing the word importance, we take the top K = 1, 2, 3 important words to replace. For the choice of the NER model, we follow the same settings as the baseline methods for fair comparison. We trained a BERT base model cased (Lafferty et al., 2001; Devlin et al., 2018) finetuned on both *CoNLL-2003* and *OntoNotes* datasets.

4.4 Benchmarking Results

4.4.1 Comparison against baselines

In Table 3, for the *CoNLL-2003* dataset, we compare the performance of our method against the five adversarial attack strategies within the SeqAttack framework. Prior to the attack, we fine-tuned our model to obtain the same F1 score before attack to align with the numbers reported using SeqAttack (original F1 score of 98%)

From Table 3, we observe that in comparison to both Bert-Attack and DeepWordBug-II_{30}, our method modifies the tokens in the original sentence at a similar level around 22%. However, the results indicate that our method achieves about 10% higher attack success rate than Bert-Attack. Also in comparison to DeepWordBug-II_{30}, we notice that our method attains a 12% lower F1 score and 24% higher on attack rate.

These significant improvements in F1 score and attack rate while having similar modification rates are attributed to the following reasons, a) our unique approach of combining effective disentangled representation learning with IG-based targeted word attributions, and b) using semantic similarity based guardrails while generating adversarial examples.

In Table 4, for the *OntoNotes* dataset, we compare the performance of RockNER and our method under three different attack settings, namely, a) context-only attacks, b) entity-only attacks and c) context + entity attacks. The second column in this table represents the baseline F1 scores on the original test set before the attack was conducted. For both RockNER and our method, successful attacks would result in lowering the F1 score compared to the baseline F1 score.

Note that RockNER chooses semantic-rich words and randomly masks tokens (3 max) in context-only attack, and replaces all named entities in entity-only attack. Different from RockNER, we compute disentangled representations and replace the most important context words (3 max) for context-only attack and the most important entity words (3 max) for entity-only attack.

Comparing the F1 scores under attack, our method is much more effective than the baseline method on all three types of attacks. More importantly, we observe that the F1 score for our method is $6\% \sim 8\%$ lower than the RockNER. This demonstrates that our method is better at selecting and replacing the most important words which enables it to generate more effective adversarial examples.

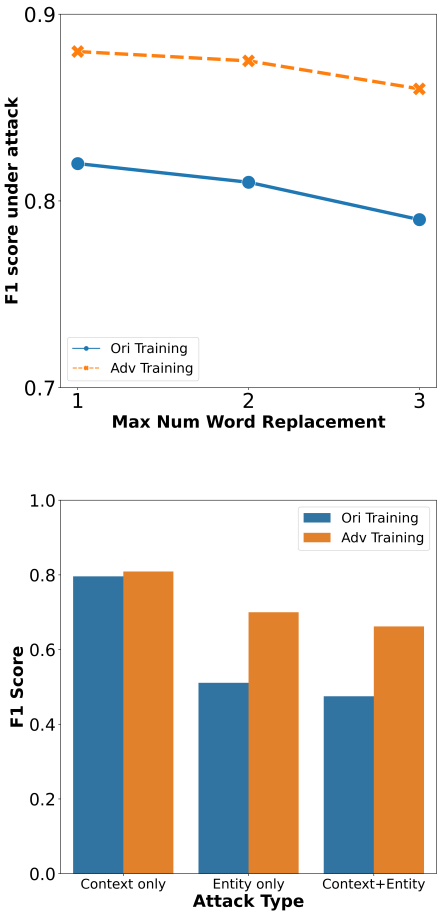

Figure 4: F1 score under attack before and after adversarial training. The upper figure shows the results on *CoNLL-2003* dataset under different levels of modification rate when generating adversarial examples. The lower figure shows the results on *OntoNotes* dataset under context-only attack, entity-only attack, and context+entity attack.

4.4.2 Case Studies

We show the generated examples from *OntoNotes* dataset using the RockNER baseline method and our method in Table 6. The sentences on the left are from the original test set and the blue texts are correctly predicted named entities by the NER model. The sentences in the middle are the generated adversarial examples using the RockNER baseline method. The sentences on the right are the generated adversarial examples following the workflow in Figure 1. The *italic* words are selected to be replaced from the original sentences. The red labels are wrong predictions by the NER model after the adversarial attacks ([O] means *Not an Entity*).

In this experiment, we allow at most one word replacement and it could either be context or a named entity word. From the output we can see that, using our method, the replacement of one word effectively causes the NER model to make wrong prediction.

From Table 6, we observe that in comparison to RockNER, the adversarial examples from our method are more semantically similar to the original test sentence. This can be attributed to the fact that our method synthesizes diverse and effective adversarial examples while preserving textual similarity. (keeping the modification rate low).

4.4.3 Adversarial Training

After generating adversarial examples from the training set using the workflow depicted in Figure 1, we conduct adversarial training on the trained NER models. Under different levels of modification rates and attack types on two datasets, we compare F1 score before and after adversarial training. As illustrated in Figure 4, we can see that adversarial training improves model robustness under different attacks settings. In addition, results show that the higher the modification rate during adversarial example generation, the higher the improvement in NER models' F1 score after adversarial training compared to the original training. For adversarial examples that allow maximum 3 word replacement on *CoNLL-2003* dataset, F1 score under attack increases about **8**% after adversarial training. F1 score under attack on *OntoNotes* dataset achieves about **18**% improvement after adversarial training. Therefore, our method is able to generate effective adversarial examples and can be used to improve the model's robustness.

5 Conclusions

In this work, we propose a novel adversarial attack for NER models which can use word attributions to identify words for substitution, and then generate adversarial examples. Our method builds upon the idea of trying to learn a balanced representation by disentangling context and entity latent representations. Applying disentanglement before computing Integrated Gradients word attributions aids in ensuring that we are able to synthesize a diverse set of adversarial examples. Experimental results on two benchmark datasets demonstrate that our method significantly outperforms baseline methods by generating more effective and diverse adversarial examples.

6 Limitations

In the current design, our attack method only shows effectiveness with disentanglement, Integrated Gradient function as word attribution, and USE as sentence encoder. Future avenues to investigate for us mainly include building an end-to-end framework to synthesize attacks for NER models where the word attributions, semantic similarity scoring and attack granularity are pursued as potential paths within a pipeline to identify the optimal attack. This will help in reducing the dependence on specific choices as done in our current work significantly.

7 Ethical Consideration

We acknowledge that our work is aligned with the *ACL Code of the Ethics* [1] and will not raise ethical concerns. We do not use sensitive datasets/models that may cause any potential issues/risks.

References

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Sudeshna Das and Jiaul Paik. 2022. Resilience of named entity recognition models under adversarial attack. In *Proceedings of the First Workshop on Dynamic Adversarial Data Collection*, pages 1–6.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Chris Donahue, Mina Lee, and Percy Liang. 2020. Enabling language models to fill in the blanks. *arXiv preprint arXiv:2005.05339*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2017. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751*.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56. IEEE.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spacy: Industrial-strength natural language processing in python.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. *arXiv preprint arXiv:1804.06059*.

Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. 2020. Captum: A unified and generic model interpretability library for pytorch.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Qi Lei, Lingfei Wu, Pin-Yu Chen, Alex Dimakis, Inderjit S Dhillon, and Michael J Witbrock. 2019. Discrete adversarial attacks and submodular optimization with applications to text classification. *Proceedings of Machine Learning and Systems*, 1:146–165.

James Melville Leland McInnes, John Healy. 2020. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv: 1802.03426*.

Bill Yuchen Lin, Wenyang Gao, Jun Yan, Ryan Moreno, and Xiang Ren. 2021. Rockner: A simple method to create adversarial examples for evaluating the robustness of named entity recognition models. *arXiv preprint arXiv:2109.05620*.

Shengcai Liu, Ning Lu, Cheng Chen, and Ke Tang. 2021. Efficient combinatorial optimization for word-level adversarial textual attack. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:98–111.

Charles Marx, Richard Phillips, Sorelle Friedler, Carlos Scheidegger, and Suresh Venkatasubramanian. 2019. Disentangling influence: Using disentangled representations to audit model predictions. *Advances in Neural Information Processing Systems*, 32.

John X Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. *arXiv preprint arXiv:2005.05909*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

[1] https://www.aclweb.org/portal/content/acl-code-ethics

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of nlp models with checklist. *arXiv preprint arXiv:2005.04118*.

Erik F Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Walter Simoncini and Gerasimos Spanakis. 2021. Seqattack: On adversarial attacks for named entity recognition. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 308–318.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR.

R Weischedel, M Palmer, M Marcus, E Hovy, S Pradhan, L Ramshaw, N Xue, A Taylor, J Kaufman, M Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. linguistic data consortium, philadelphia, pa (2013).

Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2019. Word-level textual adversarial attacking as combinatorial optimization. *arXiv preprint arXiv:1910.12196*.

Yicheng Zou, Hongwei Liu, Tao Gui, Junzhe Wang, Qi Zhang, Meng Tang, Haixiang Li, and Daniel Wang. 2022. Divide and conquer: Text semantic matching with disentangled keywords and intents.

A Example Appendix

A.1 Model Queries Comparison

We compare our attack performance with CLARE under SeqAttack framework. CLARE is a word-level attack technique which generates highest-scoring candidate sentences from replacing, inserting, and merging new words into the original sentences. According to their experimental results in the paper, it reaches F1 score 79% and attack success rate 37% whereas our attack method gets F1 score 79% and attack success rate 54%. CLARE attack allows at most **512** model queries. It generates all possible new sentences (usually 1000+) and checks if each new sentence attacks the NER model successfully. The attack success rate is very low. Among the successful attack sentences, it takes on average about **33** model queries (33 new sentences) to reach a successful attack. Compared with our method, we only allow at most **10** candidate sentences generated from each original sentence. Then we check if any sentence is a successful adversarial example which is much less than the baseline method.

Some example output from CLARE: **Attacking sample**: Japan began the defence of their Asian Cup title with a lucky 2-1 win against Syria in a Group C championship match on Friday .

AttackedText: "Japan 's began the defence of their Asian Cup title with a lucky 2-1 win against Syria in a Group C championship match on Friday ."

AttackedText: "Japan </s> began the defence of their Asian Cup title with a lucky 2-1 win against Syria in a Group C championship match on Friday ."

AttackedText "Japan A began the defence of their Asian Cup title with a lucky 2-1 win against Syria in a Group C championship match on Friday ."

AttackedText: "Japan a began the defence of their Asian Cup title with a lucky 2-1 win against Syria in a Group C championship match on Friday ."

AttackedText: "Japan ai began the defence of their Asian Cup title with a lucky 2-1 win against Syria in a Group C championship match on Friday ."

AttackedText: "Japan ain began the defence of their Asian Cup title with a lucky 2-1 win against Syria in a Group C championship match on Friday ."

AttackedText "Japan ama began the defence of their Asian Cup title with a lucky 2-1 win against Syria in a Group C championship match on Friday ."

AttackedText "Japan an began the defence of their Asian Cup title with a lucky 2-1 win against Syria in a Group C championship match on Friday ."

AttackedText "Japan and began the defence of their Asian Cup title with a lucky 2-1 win against Syria in a Group C championship match on Friday ."

AttackedText "Japan ans began the defence of their Asian Cup title with a lucky 2-1 win against Syria in a Group C championship match on Friday ." (All generated examples failed to attack)

From the outputs using CLARE, we can see that their attack strategy is to select multiple indices in the sentence for word-level operations. In the attacking sample, the attack method iteratively inserting random words between *Japan* and *began*. These random insertion incurs syntax errors in the

generated sentences. Comparing with this baseline, our method utilizes word attribution to find the most important word in the sentence more efficiently. In addition, with the help of PLMs and knowledge bases, we are able to replace with more reasonable words.

B Ablation Studies

B.1 Without disentanglement

To test the effectiveness of our proposed method, we compare the attack performance of the NER model without disentanglement. On CoNLL-2003 dataset, our ablation studies demonstrate that without the disentanglement step, the F1 score is 86% whereas the F1 score is 82% with disentanglement. The attack success rate is 25% while it is 36% with the disentanglement step.

B.2 Other word selection methods

To examine the effectiveness of Integrated Gradient, we also tested selecting words randomly for substitution. Under the same level of word modification rate, the attack success rate of random word selection is 5% less than using IG on CoNLL-2003, and 10% less than using IG on OntoNotes.

B.3 RockNER on CoNLL-2003

To compare the performance of our framework and the baseline methods, we apply RockNER to the CoNLL-2003 dataset. Their method achieved an attack success rate of 54% with a modification rate of 29%, while to reach the same level of attack success rate, our modification rate was 22%. After the adversarial training on the CoNLL-2003 dataset, the F1 under attack had 5% improvement with BertAttack while our framework had 8% improvement.