CycleNER: An Unsupervised Training Approach for Named Entity Recognition

Andrea Iovine*
University of Bari Aldo Moro, Italy
andrea.iovine@uniba.it

Anjie Fang
Amazon.com, Inc., USA
njfn@amazon.com

Besnik Fetahu
Amazon.com, Inc., USA
besnikf@amazon.com

Oleg Rokhlenko
Amazon.com, Inc., USA
olegro@amazon.com

Shervin Malmasi
Amazon.com, Inc., USA
malmasi@amazon.com

ABSTRACT

Named Entity Recognition (NER) is a crucial natural language understanding task for many down-stream tasks such as question answering and retrieval. Despite significant progress in developing NER models for multiple languages and domains, scaling to emerging and/or low-resource domains still remains challenging, due to the costly nature of acquiring training data. We propose CycleNER, an *unsupervised* approach based on cycle-consistency training that uses two functions: (i) *sentence-to-entity* – S2E and (ii) *entity-to-sentence* – E2S, to carry out the NER task. CycleNER does not require annotations but a set of sentences with no entity labels and another independent set of entity examples. Through *cycle-consistency* training, the output from one function is used as input for the other (e.g. S2E → E2S) to align the representation spaces of both functions and therefore enable unsupervised training. Evaluation on several domains comparing CycleNER against supervised and unsupervised competitors shows that CycleNER achieves highly competitive performance with only a few thousand input sentences. We demonstrate competitive performance against supervised models, achieving 73% of supervised performance without any annotations on CoNLL03, while significantly outperforming unsupervised approaches.

CCS CONCEPTS

• **Computing methodologies → Information extraction**; *Natural language generation*; **Unsupervised learning**.

KEYWORDS

natural language processing, named entity recognition, cycle-consistency training, unsupervised training

ACM Reference Format:
Andrea Iovine, Anjie Fang, Besnik Fetahu, Oleg Rokhlenko, and Shervin Malmasi. 2022. CycleNER: An Unsupervised Training Approach for Named Entity Recognition. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3485447.3512012

* This research was done during an internship at Amazon.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France

© 2022 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-9096-5/22/04.
https://doi.org/10.1145/3485447.3512012

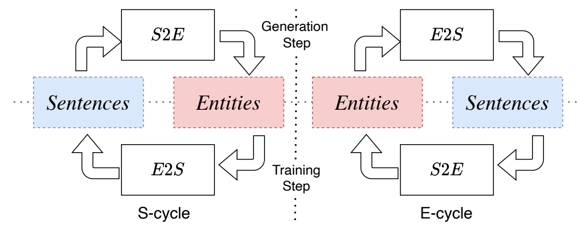


Figure 1: Overview of the two CycleNER tasks.

1 INTRODUCTION

Named Entity Recognition (NER) is a core NLP task, being applied in Web search [4], conversational agents [35], and Relation Extraction [1], with increasing adoption in specialized domains like medicine [19, 30], analysis of historical collections [9], etc.

NER approaches are typically trained in a fully supervised manner. The NER training data consists of token-level annotations, where each token is annotated according to a NER class taxonomy (e.g. `PER`, `ORG`, `O` etc.). While there are significant annotation efforts for popular domains like *news* [23], annotations for specialized domains, e.g. *medicine*, are difficult to obtain due to: (i) annotator training being complex and time consuming, with domain knowledge acquisition being key (e.g. that *"HKE6"* is a `Gene`), and (ii) token-level annotations from diverse domains at scale being costly.

Unsupervised NER methods can help alleviate some of these requirements. Existing unsupervised approaches are based on complex hand-crafted rules, or rely on entity information from pre-existing lexicons or knowledge bases (e.g. [3, 14]). However, it is challenging to create rules and lexicons for entities from special domains, and even harder for low-resource languages. In this paper, we propose CycleNER, an *unsupervised* approach for training NER models using unannotated sentences, and very small named entity samples that are independent from the sentences[1]. CycleNER has two components: (i) *sentence-to-entity* (S2E), and (ii) *entity-to-sentence* (E2S). S2E extracts entities given a sentence, while E2S generates a sentence given input entities. S2E and E2S are implemented using sequence-to-sequence transformer architectures since this allows bi-directional conversion between sentences and entities. These two components are jointly trained via two cycles (as shown in Figure 1): (1) sentence learning cycle (S-cycle) and (2) entity learning cycle

[1] In fact, entity samples can be easily acquired through sources such as knowledge bases.

(E-cycle). Specifically, sentences are the input and output for the S-cycle, while entities are the same for the E-cycle. In each cycle, one component first generates the inputs and the other component is then trained using these generated inputs. In this way, two components are trained effectively and their representation spaces are aligned. Compared to the traditional NER training approach, cycle-consistent training used unlabeled data and is therefore unsupervised. This data can be much cheaper to obtain than traditional supervised datasets.

We experiment with several NER benchmarks, i.e. CoNLL03, WNUT17, OntoNotes 5.0, and BioCreative II BC2GM. We first verify that the sequence-to-sequence architecture, i.e. T5 [22], allows us to effectively extract entities from sentences. Since CycleNER implements NER by two-cycle training, which is different from the traditional token classification, we experiment with different training strategies. We evaluate the model by setting different numbers of sentences and entity examples, and compare it to SOTA NER results. Results show that our CycleNER can achieve competitive performance compared to supervised approaches. For example, when using a thousand entity examples, CycleNER can achieve 0.686 (72.7% of SOTA) and 0.613 (66% of SOTA) in CoNLL03 and OntoNotes 5.0, respectively. Our contributions are summarized as follows:

- We propose CycleNER, a novel unsupervised training approach.
- We implement CycleNER using two encoder-decoder architectures, E2S and S2E.
- We study how to effectively train NER using CycleNER.
- We show through an ablation study that our approach increases performance as more sentence or entity examples are provided.
- We compare our approach against several supervised and unsupervised baselines, achieving competitive results.

2 RELATED WORK

Supervised NER: NER is typically cast as a supervised learning task, with most of the state-of-the-art approaches relying on deep recurrent models [10, 21], convolution based [16], or pre-trained transformer architectures [8, 32]. These models achieve highly satisfactory NER performance on typical benchmarking datasets like CoNLL. Yet, the results achieved on CoNLL do not transfer across domains [17], and supervised data from target domains is necessary.

While NER is typically performed through *sequence labelling*, recent approaches have cast this problem as a sequence-to-sequence (*seq2seq*) task. Zhu et al. [36] use a Bi-LSTM model to encode sentences and an LSTM+CRF to generate the output entities from an input sentence. Compared against several supervised baselines, the approach has proven to be effective. Similarly, Straková et al. [26], tackle the problem of nested NER, where for a token all possible entity labels are generated. Arguably, recent advances in pre-training seq2seq transformer models, like T5 [22] or BART [12], can be used to perform NER more effectively than recurrent models (LSTMs).

Unsupervised NER: Unsupervised approaches typically rely on hand-crafted rules and pre-obtained lexicons. Etzioni et al. [3] extract entities according to syntactic pattern matches. Zhang and Elhadad [34] propose an unsupervised approach that is applied on the medical domain. The approach first obtains seed entities from an external source, then identifies entities from sentences through chunking and using inverse document frequency. Similarly, Liang et al. [14] propose to generate distant labels using knowledge bases and use these labels to improve supervised NER training. Liu et al. [15] also use a knowledge base to train a NER model (KALM) by identifying whether a word in a sentence is from knowledge base or a general dictionary. While these methods highly rely on external knowledge bases, CycleNER aims at using very limited entity samples, without the need of external resources. In [28], a neural Hidden Markov Model (NeuralHMM) is proposed for token annotation. It estimates the probability distribution of the latent classes using the Baum-Welch algorithm. To do so, it relies on a set of lexical, morphological and syntactic features extracted using neural networks. Morphological features are extracted using CNN, whereas the sentence's context is captured using LSTMs. This approach is shown to be effective for POS tagging. Both POS tagging and NER can be cast as sequence labeling problems, and therefore we consider this approach as an unsupervised baseline.

Cycle-consistency Training: First introduced in neural machine translation (NMT) [11, 18, 24], cycle-consistency training has been applied to align the latent spaces of auto-encoders trained on different languages, such that for a few seed words and their corresponding translations, their representations are aligned. This allows to train NMT models without parallel datasets. Recently, Guo et al. [5] proposed CycleGT, which jointly learns graph-to-text and text-to-graph tasks. CycleGT is trained using non-parallel data, consisting of textual snippets and graph triples. To solve the multiple mapping problem between text and graph modalities, Guo et al. [6] propose a conditional variational auto-encoder to transform the surjective function into an implicit bijection.

Contrary to previous works, our approach does not require adversarial training or denoising auto-encoder strategies. Instead, we exploit pre-trained transformer models as a means to regularize the training of CycleNER. Furthermore, contrary to Mohiuddin and Joty [18], our cycle-consistency training does not use two latent spaces, but rather, the output of each models (S2E or E2S) is fed as input to each other to generate intermediate sentences or entity sequences.

Our approach transfers the intuitions from Lample et al. [10] and Guo et al. [5]. Unlike Guo et al. [5], we are not constrained on having different modalities in order to perform cycle-consistency training, since we treat NER as a text-to-text generation task.

3 CYCLE-CONSISTENCY NER

We formulate CycleNER as an unsupervised cycle-consistency learning problem, outlining the intuition, the required data and tasks.

Cycle-consistency learning involves simultaneously learning a forward and inverse transformation of data. This approach can be applied to unannotated non-parallel data [37] to learn mapping functions in an unsupervised setting. We propose to apply this framework for NER, using non-parallel entities and sentences, and training generative models to transform sentences to entities, and vice versa.

Unsupervised NER Data: CycleNER relies on non-parallel data, i.e. a set of sentences $S = \{s_1, \ldots, s_n\}$, where each sentence s_i can mention zero or more entities, and a set of entity sequences Q=$\{\langle e_{1,1} \ldots, e_{1,k}\rangle, \ldots, \langle e_{m,1}, \ldots, e_{m,l}\rangle\}$, where each sequence $\langle e_{i,1}, \ldots, e_{i,k}\rangle$ consists of zero or more entities. Specifically, we use an entity sequence to represent entities contained in a sentence. The two sets S and Q are unannotated, i.e. S does not have entity annotations and

Organization Miscellaneous Miscellaneous
s: *EU rejects German call to boycott British lamb*

q: *EU* <sep> *Organization* <sep> *German* <sep> *Miscellaneous* <sep> *British* <sep> *Miscellaneous*

Figure 2: Entity sequence q format from sentence s.

Q is an independent set of entity sequences. The only prerequisite is that the entity distribution appearing in S and Q have some overlap.

Training Tasks: Given S and Q, in CycleNER we define two main tasks in CycleNER:

(1) Sentence-to-Entity (S2E): For a sentence $s \in S$, S2E outputs an entity sequence q'. This represents the NER task in CycleNER, e.g.:

> s: *"Australia beat Sri Lanka 2-1 in the World Series cricket match on Wednesday"*
> q': ("Australia", `LOC`), ("Sri Lanka", `LOC`), ("World Series", `ORG`)

(2) Entity-to-Sentence (S2E): For an input entity sequence $q \in Q$, E2S generates an output sentence s' containing the entities in q, e.g.:

> q: ("EU", `ORG`), ("German", `LOC`), ("British", `MISC`)
> s': *"**EU** rejects **German** call to boycott **British** lamb."*

4 CYCLENER: UNSUPERVISED NER

Figure 3 shows an overview of CycleNER, where S2E and E2S are used to implement *S*- and *E*-cycles, which ensure cycle-consistency and allow us to train them in an unsupervised manner. Next, we describe in detail S2E and E2S, and the cycle-consistency training.

4.1 Sequence-to-sequence for CycleNER

In cycle-consistency training S2E represents the inverse function of E2S, and vice versa. To meet this requirement, S2E and E2S should share the same input and output format, i.e. textual format. Moreover, the output should retain the information needed to easily reconstruct the input from it. In *sequence labeling*, a sentence is classified as a sequence of IOB tags, where the tags are neither in the same format as the input sentence, nor they retain any token information. Hence, *sequence labeling* cannot be used to implement S2E and E2S.

On the other hand, *seq2seq* can naturally fit this task, as it can be used to generate entity tokens (denoted as $q \in Q$) directly from a sentence (e.g. in [36]). First, s is passed through a *seq2seq* encoder, then the decoder generates a sequence q a token at a time. Since q contains important information about important tokens from s, it can be to do construct a sentence in E2S.

Entity Sequence Representation in CycleNER. As shown in Figure 2, a sentence can contain multiple entities. We propose to use an entity sequence, q, to represent multiple entities in a sentence in CycleNER. The entity sequence is required to contain all entity information to enable cycle-consistency training. For this, we first represent an entity as a combination of its *surface form* and *entity class* in textual form. One sentence can contain multiple entities, denoted as an entity sequence q. A special token, ⟨`sep`⟩, separates different entities in q, as well as the surface forms and class tokens within an entity $e \in q$. Figure 2 shows an example of the proposed format. In case multiple occurrences of the same entity are present in a sentence, q will contain the repeated entity as many times as it appears in the sentence. Moreover, the order of the entities in q reflects the order of them appearing in the sentence. This guarantees that S2E can extract correct entities from sentences. For example, given a sentence *"The Amazon CEO pledged to donate to the Amazon rainforest preservation efforts"*, q should be "*Amazon* ⟨`sep`⟩ `organization` ⟨`sep`⟩ *Amazon* ⟨`sep`⟩ `location`". Furthermore, keeping the entity order simplifies the reconstruction of the original sentence during CycleNER.

4.2 Cycle-Consistency Training

The objective of cycle-consistency is to ensure that S2E and E2S functions establish an effective mapping between sentences and entities. We follow the principles of Iterative Back-Translation (IBT) [5, 7] to incrementally train the two functions, namely the output of one function is used as input to train the other function. We denote with θ and ϕ the parameters of S2E and E2S, respectively. The training is divided into two main cycles: S-cycle and E-cycle.

S-cycle Training. In this cycle, we generate first a synthetic set of intermediate entity sequences by applying S2E to the input S, namely $S2E(S) \rightarrow Q'$, highlighted by the red box in Figure 3. The synthetic training tuples $\langle S, Q' \rangle$ are used to train E2S in a supervised manner. Specifically, in this cycle E2S is trained to generate synthetic sentences S' that are similar to S (highlighted in the blue box). For instance, in Figure 3 (a), in the first step S2E receives an input sentence s, and produces the output entity sequence q'. In the second step, E2S receives the synthetic q' and generates back a synthetic sentence s' that ideally should mimic the original input sentence s, as shown in the example below:

- s: "I'm not retiring, Duran told Reuters."
- q': "*Duran* ⟨`sep`⟩ `person` ⟨`sep`⟩ *Reuters* ⟨`sep`⟩ `organization`"
- s': *"Duran told Reuters: We have no plans to sell the shares"*.

This concludes the S-cycle training, at the end of which, similar to Guo et al. [5] we compute the reconstruction loss, i.e. the average cross-entropy loss between the input S and the generated S':

$$\mathcal{L}_{\phi}(S, S') = -\frac{\sum_{s \in S} \sum_{i<|s|} p(s_i) \log g(s'_i)}{|s| * |S|} \tag{1}$$

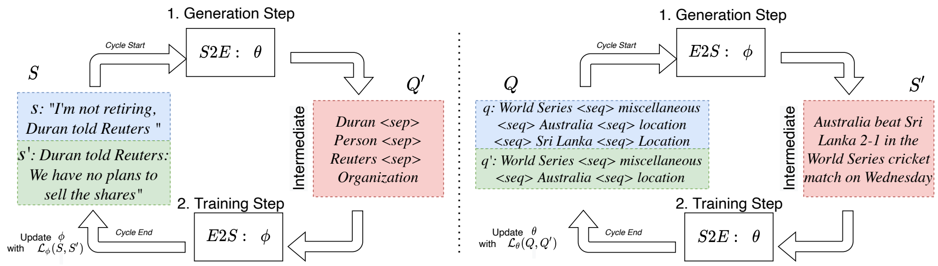
where $p(\cdot)$ and $g(\cdot)$ represent the real and predicted token probabilities. Whereas, $|s|$ represents the sentence length, s_i and s'_i are the i-th token in s and s', and $|S|$ is the number of input sentences.

E-cycle Training. In this cycle, the input are entity sequences Q, which are pushed through E2S to generate a set of synthetic sentences S', namely $E2S(Q) \rightarrow S'$, cf. highlighted boxes in red in Figure 3 (b). The E-cycle trains S2E to generate the entity sequences Q' that are similar to Q. First, for an entity sequence q a synthetic sentence s' is generated via E2S, which is then fed into S2E to generate the synthetic q', that is identical or highly similar to q:

- q: " *World Series* ⟨`sep`⟩ `miscellaneous` ⟨`sep`⟩ *Australia* ⟨`sep`⟩ `location` ⟨`sep`⟩ *Sri Lanka* ⟨`sep`⟩ `location`"
- s': *"Australia beat Sri Lanka 2-1 in the World Series cricket match on Wednesday"*
- q': "*World Series* ⟨`sep`⟩ `miscellaneous` ⟨`sep`⟩ *Australia* ⟨`sep`⟩ `location`"

Similar to S-cycle, we compute the reconstruction loss between the input entity sequences Q and the synthetic ones Q' in the E-cycle:

$$\mathcal{L}_{\theta}(Q, Q') = -\frac{\sum_{q \in Q} \sum_{i<|q|} p(q_i) \log g(q'_i)}{|q| * |Q|} \tag{2}$$



(a) S-cycle (b) E-cycle

Figure 3: Training cycles and steps in CycleNER.

where $|q|$ and $|Q|$ are the fixed sequence length (i.e. the number of surface form tokens in q) and total number of entity sequences. q_i and q'_i are the i-th token in q and q'. Note that since the entity sequence contains both the surface form and entity type, $L_\theta(Q, Q')$ reflects the loss of both the surface form and entity type.

Joint training. The two cycles are conducted iteratively. Operationally, the following steps are executed in sequence for each batch of training data:

- **S-cycle Step 1** (see Figure 3): a batch of synthetic sequences Q' is generated by S2E giving batch of training sentences S.
- **S-cycle Step 2**: The parameters of E2S are trained using synthetic training tuples $\langle Q', S \rangle$.
- **E-cycle Step 1**: A batch of synthetic sentences S' is generated by E2S giving a batch of training entity sequences Q.
- **E-cycle Step 2**: The parameters of S2E are trained using synthetic training tuples $\langle S', Q \rangle$.

S2E leverages Q to recognize entities and their contexts in S, while E2S learns to generate plausible sentences using the entries in Q. The parameter spaces of the two tasks are trained to align each other, such that from s we can generate entity sequences q, and vice versa. Cycle-consistency enforces the reconstruction of the original input sentence s from the generated entity sequence q', or generating back the entity sequence q from an output sentence s'. This ensures that S2E and E2S can learn quickly even in the beginning stage. Accordingly, E- and S-cycles are equally important for CycleNER.

In order to select the best iteration during training (stopping criterion), we use the loss of E-cycle on a development set. This choice is futher discussed and verified in Section 7.2. Finally, it is worth noting that during the *S-cycle*, the parameters θ are not updated, due to the fact that the generated synthetic outputs at the end of Step 1 are non-differentiable. Effectively, $\mathcal{L}_\phi$ can only be back-propagated to E2S. Similarly, during the *E-cycle*, ϕ is not updated, and $\mathcal{L}_\theta$ is only back-propagated to S2E. This is similar to previous applications of IBT [5].

5 DATA

5.1 NER Datasets

We consider several NER benchmarking datasets across various domains to sample sentences and entity sequences for training CycleNER. Table 1 shows an overview of the datasets. All the datasets are in English language.

CoNLL [27] is a common NER benchmark dataset, consisting of textual snippets from the *news* domain.
WNUT [2] contains textual snippets extracted from social media, e.g. Twitter. This dataset represents a challenging benchmark with novel and emerging entities.
OntoNotes [31] contains mostly sentences from news, Web data and conversational speeches. Specifically, we use the version that includes a taxonomy of 18 NER classes.
BioCreative II BC2GM [25] consists of sentences extracted from medical articles, with *genes* as the annotated named entities. This dataset is very challenging, given that the surface form of gene sequence entities can be highly complex.

Dataset	Type	# Sent	# Token	Sent Len	# Tags
CoNLL	Train	14,041	203,621	14.50	4
	Dev	3,250	51,362	15.80	
	Test	3,453	46,435	13.45	
WNUT	Train	3,394	62,730	18.48	6
	Dev	1,009	15,733	15.59	
	Test	1,287	23,394	18.18	
OntoNotes	Train	115,864	2,200,868	19.00	18
	Dev	15,680	304,701	19.43	
	Test	12,217	230,118	18.84	
BC2GM	Train	12,574	355,405	28.27	1
	Dev	2,519	71,042	28.20	
	Test	5,038	143,465	28.48	

Table 1: Dataset statistics.

5.2 Entity Sequences

To train the E2S function, we feed it named entity sequences as input. The length of the entity sequences should reflect the distribution of sequences in our sentences. We compare two methods of acquiring entity sequences. The first solution directly uses a small portion of entity sequences from the ground-truth, which ensures high data quality, and helps to test our approach in an ideal scenario. In the second solution, we select a small sample of entities from the ground truth and generate synthetic entity sequences. This represents a realistic scenario where we only have a small size of entity examples. We expect that synthetic sequences can work similarly as ground truth sequences.

1. Ground truth Entity Sequences – ***GQ***. From each sentence in the NER datasets, we extract the entity sequences. We then train E2S

Dataset	*GQ*	*SQ*
CoNLL	Kragujevac ⟨sep⟩ *location* ⟨sep⟩ Stanojlovic ⟨sep⟩ *person*	National Tennis Centre ⟨sep⟩ *location*
WNUT	Justin Timberlake ⟨sep⟩ *person* ⟨sep⟩ Beyonce ⟨sep⟩ *person* ⟨sep⟩ Until the End of Time ⟨sep⟩ *creative work*	Mayflower ⟨sep⟩ *group* ⟨sep⟩ Richard Smith ⟨sep⟩ *person*
OntoNotes	This year ⟨sep⟩ *date* ⟨sep⟩ Japan ⟨sep⟩ *geopolitical* ⟨sep⟩ US ⟨sep⟩ *geopolitical* ⟨sep⟩ Silicon Valley ⟨sep⟩ *location* ⟨sep⟩ the United States ⟨sep⟩ *geopolitical*	Ecuadorian ⟨sep⟩ *group* ⟨sep⟩ Colombia ⟨sep⟩ *geopolitical* ⟨sep⟩ November ⟨sep⟩ *date*
BC2GM	NFI proteins ⟨sep⟩ *gene* ⟨sep⟩ TG ⟨sep⟩ *gene* ⟨sep⟩ NFI - DNA ⟨sep⟩ *gene*	C protein ⟨sep⟩ *gene* ⟨sep⟩ Protein kinase C ⟨sep⟩ *gene*

Table 2: *GQ* and *SQ* entity sequence examples.

on the extracted sequences. The *GQ* entity sequence are noise-free and consist of only entity sequences that appear in real-world data.
2. Synthetic Entity Sequences – *SQ*. Starting from a set of seed entities extracted from ground-truth, we construct sequences by pairing seed entities with other entities that are semantically similar to them. We calculate a vector representation for each entity using pre-trained word embeddings [20], where each vector is the average of all word embeddings from an entity. Then, we group similar entities together into an sequence by calculating their cosine-similarity. This approach guarantees that the generated sequence can be mapped back to a plausible sentence by E2S. When creating the entity sequences, we choose the length (i.e. # of entity in an sequence) by considering the length distribution from the original training set. Our preliminary experiment shows that this step is not strictly necessary. It ensures that S2E can generate entity sequences of different sizes. This approach can be used to generate sequences for a real application with a small size of entity examples. Compared to traditional parallel NER annotations, these entity examples can be easy to access.

5.3 CycleNER Training Data

CycleNER requires a set of sentences and a set of entity sequences. From the different NER datasets, we construct varying sets of training data, where we vary the number of sentences and entity sequences used for training. For the sake of clarity, we introduce a naming convention to distinguish the different dataset configurations. Namely, `Dataset/10k/10k`$_{SQ}$, where the first portion represents the dataset, followed by the number of sentences and entity sequences in *thousands*, and the means by which we extract the entity sequences. Table 3 shows the different configurations we use for training and development data, where we vary the amount of sentences and entity sequences we use to train CycleNER.

6 EXPERIMENTAL SETUP

This section describes the baselines, and the setup of our approach. Furthermore, we explain evaluation scenarios that we want to validate in our experimental evaluation.

6.1 Baselines and Our Approach

CycleNER: For our approach the S2E and E2S represent seq2seq models. We implement the two functions using T5 [22] pre-trained transformer models.
BERT: This represents a competitive supervised baseline, and can be considered as the upper bound for NER performance of unsupervised models. We fine-tune a pre-trained BERT [8] for NER.

Dataset	Configuration	Set
CoNLL	/1k/1k$_{GQ}$; /2k/1k$_{GQ}$; /14k/14k$_{SQ}$	train
	/3.2k/1k$_{GQ}$; /3.2k/2k$_{SQ}$	dev
WNUT	/1k/1k$_{GQ}$; /3.4k/1k$_{GQ}$; /3.4k/3.4k$_{SQ}$	train
	/1k/500$_{GQ}$; /1k/500$_{SQ}$	dev
OntoNotes	/5k/5k$_{GQ}$; /116k/2k$_{GQ}$; /116k/116k$_{SQ}$	train
	/15.7k/5k$_{GQ}$; /15.7k/5k$_{SQ}$	dev
BC2GM	/1k/1k$_{GQ}$; /12.5k/1k$_{GQ}$; /12.5k/12.5k$_{SQ}$	train
	/2.5k/1k$_{GQ}$; /2.5k/1k$_{SQ}$	dev

Table 3: Dataset configurations with varying number of sentences and entity sequences.

NeuralHMM: This represents an unsupervised baseline, which trains a neural Hidden Markov Model using sentences as input only. NeuralHMM requires only sentences for training, and it does not require entity information for training. Its output space is a set of latent classes, the number of which can be set as a learning parameter. Additionally, a mapping strategy is required to link the latent classes to entity classes. To do this, the co-occurrence between each latent class and entity class in the test set is measured, and the most frequently co-occurring latent class with an entity class is assigned.
Lexical Matcher: We extract the named entities from the training set and their corresponding type, which then using a lexical matcher (surface form match) are used to identify entities in the test set. This represents a basic unsupervised baseline, and fails for entities with ambiguous surface forms.
BERT-Matcher: We also train a weakly-supervised BERT model using external entity knowledge, a common approach in the literature. We employ a similar method as Meng et al. [17] to first generate gazetteer data (3.9m entities) for CoNLL and WNUT. We then apply the lexical matcher and the gazetteer data to create weakly-annotated parallel NER training data from CONLL and WNUT sentences. We train our NER models using BERT with these datasets. Specifically, we check whether a given sentence contains an entity entry from the gazetteer and create parallel entity annotation for the sentence. However, this gazetteer data is noisy, e.g. an entity entry can belong to multiple entity classes. Therefore, we do not use the gazetteer for CyclerNER.

6.2 Evaluation Scenarios

We empirically assess CycleNER under the following scenarios:

	BiLSTM	T5	LUKE [32]	T-NER [29]
CoNLL	0.551	0.913	0.943	-
WNUT	-	0.552	-	0.585

Table 4: NER performance of supervised models for NER as a sequence generation task, compared against the SOTA for each dataset.

Scenario 1: Can the NER task be cast as a sequence generation task using our proposed sequence format (see Figure 2)?
Scenario 2: Does unsupervised training work? What is the relation between the reconstruction loss and NER F1?
Scenario 3: How does the number of sentences and entity sequences impact CycleNER?
Scenario 4: How does CycleNER fare in contrast to supervised NER models?

To answer evaluation scenarios (1)–(3) we use the CoNLL and WNUT datasets. For scenario (4) we use all the datasets, namely the different configurations (cf. Table 3). We measure NER performance using the micro-averaged F1 score.

7 RESULTS

7.1 NER as a Sequence Generation

In the first evaluation scenario, we assess how *seq2seq* models address NER. As mentioned in Section 4.1, given a sentence, the *seq2seq* model outputs the mentioned entities with their types. We mainly verify the suitability of casting NER as a sequence generation task, given that it is one of the fundamental principles in training CycleNER in an unsupervised manner.

Table 4 shows the performance of a BiLSTM and T5 model trained on the entire dataset. The training is done in a supervised manner. Comparing to BiLSTM, we note that T5 has superior performance on both datasets[2]. This is attributed mainly to the sophisticated architecture of T5 and its extensive pre-trained knowledge, which allows it to better capture contextual information.

When comparing the T5 sequence model to the state-of-the-art results, T5 achieves very similar results. For both CoNLL and WNUT, T5 has only a ~3% drop in terms of F1. These results validate our hypothesis that NER can be cast as a sequence generation task and our entity sequence representation is effective.

7.2 CycleNER Training Behavior

In the standard supervised setting, training is stopped whenever a given loss function (e.g. cross-entropy) converges to a minimum. However, in CycleNER, training is done in an unsupervised manner. Hence, determining when the model, namely the two functions S2E and E2S, are fully trained is not trivial. To determine when training has converged, we analyze the relation of the two reconstruction losses that CycleNER minimizes: $\mathcal{L}_\theta$ from the E-cycle, and $\mathcal{L}_\phi$ from the S-cycle. As the two cycles are used to train the two functions, S2E and E2S, convergence in both cases is important to have stable and optimal performance for CycleNER.

[2]For WNUT, BiLSTM produces poor results, hence, we omit the results from the table.

(a) Varying number of entity sequences

	#Q	100	1k	1.5k	2k	3k	14k
CoNLL (#S = 14k)	*GQ*	0.619	0.814	0.823	0.842	0.852	0.885
	SQ	0.584	0.673	0.637	0.667	0.676	0.686
	#Q	500	1k	1.5k	3.4k		
WNUT (#S = 3.4k)	*GQ*	0.327	0.338	0.316	0.332		
	SQ	0.349	0.320	0.321	0.336		

(b) Varying the number of sentences

	#S	1k	1.5k	2k	3k	14k
CoNLL (#Q = 1k)	*GQ*	0.804	0.797	0.825	0.823	0.814
	SQ	0.609	0.616	0.613	0.628	0.673
	#S	500	1k	1.5k	3.4k	
WNUT (#Q = 500)	*GQ*	0.282	0.323	0.335	0.338	
	SQ	0.251	0.292	0.269	0.320	

Table 5: F1 performance when varying the number of training (a) entity sequences and (b) sentences. #S is the number of training sentences, #Q is the number of training entity sequences.

We assess the best stopping criterion by training CycleNER separately on the CoNLL and WNUT datasets, and consider both entity sequence generation approaches (i.e. *GQ* and *SQ*). Namely, we use the following configurations for training: $\texttt{CoNLL/14k/1k}_{GQ}$, $\texttt{CoNLL/14k/1k}_{SQ}$, $\texttt{WNUT/3.4k/1k}_{GQ}$, and $\texttt{WNUT/3.4k/1k}_{SQ}$. The relatively small size of entity sequences allows us to quickly conduct experments.

Figure 4 shows the reconstruction losses at different epochs for the S- (blue line) and E- (orange line) cycles on the development set (cf. Table 3). Alongside the loss values we plot the corresponding F1 scores. These high F1 scores (e.g. ~0.8 for CONLL and ~0.4 for WNUT) suggest that CycleNER is effective during training.

Figure 4 shows a clear relationship between the loss computed in the E-cycle and F1. For CONLL, we obtain a *high negative correlation* as measured through Pearson's correlation coefficient, with $\rho = -0.81$ for *GQ*, and with $\rho = -0.87$ for *SQ*. For WNUT, we observe similar correlations, with a high negative correlation for *GQ* with $\rho = -0.73$, whereas a moderate correlation is observed for *SQ* with $\rho = -0.59$. Contrary to the E-cycle, there is no clear relationship between the S-cycle loss and F1, with an average correlation coefficient of $\rho = 0.41$ over CoNLL and WNUT, respectively for both *GQ* and *SQ*. This is because that the reconstructed sentence in the S-cycle can be different from the original one although they contain the same entities.

We conclude that the E-cycle loss can be used to to guarantee a optimal NER performance. Therefore, we use it as the stopping criterion for all subsequent tests described in Sections 7.3 and 7.4.

7.3 Impact of Training Data Size

In this evaluation scenario, we assess the impact of the training data used for CycleNER. Table 5 shows the impact of training data for CoNLL and WNUT datasets, where we vary either both the number of sentences (*S*) or entity sequences (*Q*) used for training.

Number of Entity Sequences. Table 5 (a) shows the performance of CycleNER when trained on a fixed number of sentences,

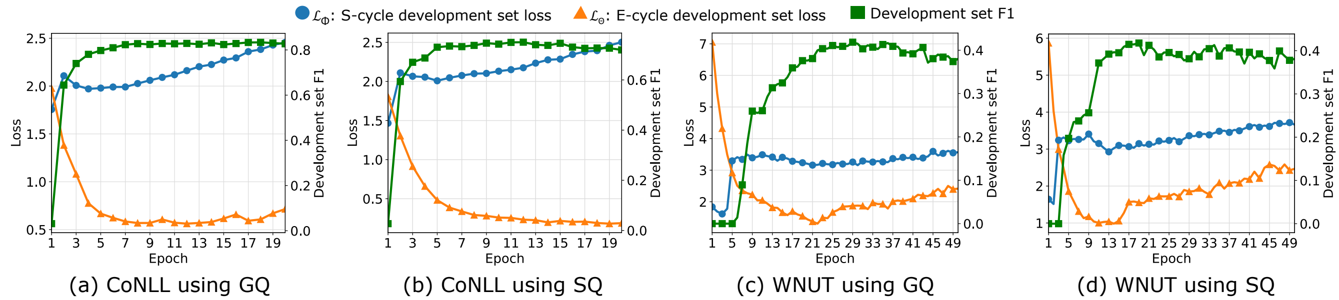


Figure 4: Development set loss of $\mathcal{L}_\theta$ and $\mathcal{L}_\phi$ and F1 score per epoch.

namely 14k for CoNLL, and 3.4k for WNUT, while varying the number of entity sequences. At a high level we note the good performance of CycleNER using very small amounts of entity knowledge.

For CONLL, when using 100 *GQ* entity sequences, the model performs modestly, with $F1 = 0.619$. However, when the number of entity sequences is increased to 1k, then we see a nearly 20% absolute improvement in terms of F1. Adding more entity sequences results in gradual performance increase, however, the rate of improvement is slightly lower. This finding is interesting, where for sentences that are of fairly similar text genres, with relatively little data, we can converge to the top performance using CycleNER. The performance of *SQ* is lower compared to *GQ*, and although more sequences enable better performance, this is not always the case, as it can be seen between 1k and 1.5k sequences. The difference between *GQ* and *SQ* can due to two main reasons. First, pre-trained embedding might not effectively capture the semantic meaning of an entity and find its similar ones. Second, given that CoNLL contains sentences from news corpora, the set of entities that co-occur in a sentence is often determined by newsworthiness factors, and are not correlated with entity relatedness solely.

Contrary to the CoNLL dataset, for WNUT, the performance difference between entity sequences generated according to *GQ* or *SQ* is marginal. A possible explanation for this difference w.r.t CoNLL, is that WNUT consists of text snippets coming from social media, and thus, the set of entities that co-occur in a sentence is much more diverse, and less controlled as in news media.

For both datasets, CycleNER is able to maintain reasonable effective NER performance with a small size of entity examples. In particular, CycleNER achieved 0.34 F1 (vs. T-NER's 0.585 [29]) using only 500 *SQ* for WNUT, where entities are from social media and are highly complex.

Number of Sentences. Table 5 (b) shows the performance of CycleNER when trained on a fixed set of entity sequences, namely 1k and 500, for CoNLL and WNUT, respectively, while at the same time varying the number of input sentences.

As in the case of varying entity sequences, adding sentences translates into better NER performance. For CONLL, the difference is nearly 2% absolute points improvement for *GQ*, and 7% for *SQ* sequences. While, in the case of WNUT, the improvements are with 5% and 7%, for *GQ* and *SQ*, respectively.

Although the gains are significant given the scale of the datasets, the improvements are more moderate than when varying entity sequences. This is intuitive, considering that in Table 5 (a) we show that with 1k entity sequences, the model achieves a performance that is close to its peak. At the same time, we note that the impact of additional sentences for training is much larger for *SQ*.

In summary, more data helps overall to improve the model's performance. While, when comparing additional sentences or entity sequences, we note that more entity sequences are more beneficial for CycleNER, which allows the model to better learn the NER task.

7.4 Supervised vs. Unsupervised Approach

Table 6 shows the results of the different NER approaches defined in Section 6.1. Apart from BERT and BERT-Matcher, which perform NER in the standard token classification setting, the rest of the approaches are unsupervised. The SOTA row reports the performance for each dataset from existing work. We also report the results obtained by KALM on the CoNLL dataset, as reported in [15].

It is worth noting that NeuralHMM is trained only using sentences, whereas CycleNER is trained with a fairly larger amount of entity sequences for *SQ*, while for *GQ* this amount is much smaller. More specifically, for CoNLL we use 1k *GQ* sequences (representing 7% of the total sequences), 1k for WNUT (29% of the original size), 5k for Ontonotes (4.3% of the original size), and 1k for BC2GM (8% of the original size).

Performance Comparison. In all cases, BERT achieves the best performance. This is intuitive given that it uses annotated data at the token level, and thus, the loss is optimized at the token level, allowing the model to achieve optimal performance. However, as mentioned in the motivation of this work, obtaining such annotated data is not always feasible and can be costly. On the contrary, BERT-Matcher (trained using gazetteer-based weakly-annotated data) performs worst. This is because the weakly-annotated data is too noisy.

When comparing the performance of CycleNER and BERT across datasets, we note that the differences are not high. For instance, when CycleNER is trained on $\texttt{CoNLL/2k/1k}_{GQ}$, the difference is 3.5%, which when considered that our approach is unsupervised presents remarkable results. The gap is higher, with 17.4%, when we use $\texttt{CoNLL/14k/14k}_{SQ}$ for training. As noted in Section 7.3, co-occurrence of entities in the news domain is determined by the newsworthiness of entities. Hence, *SQ* sequences, may introduce infrequent sequences in news.

We note similar observations across the different datasets, such as WNUT, OntoNotes, where the gap is 7%, and 13.5%, respectively. For both datasets, the gap between *GQ* and *SQ* entity sequences is

Approach	CoNLL (CoNLL)	P	R	F	WNUT (WNUT)	P	R	F	OntoNotes (OntoNotes)	P	R	F	BC2GM (BC2GM)	P	R	F
SOTA		Yamada et al. [33]		0.943	Ushio and Camacho-Collados [29]			0.585		Li et al. [13]		0.927		Smith et al. [25]		0.872
BERT	/1k	0.844	0.876	0.860	/1k	0.482	0.369	0.418	/5k	0.756	0.823	0.788	/1k	0.651	0.713	0.681
BERT-Matcher	full	0.147	0.415	0.217	full	0.029	0.162	0.050	-	-	-	-	-	-	-	-
NeuralHMM	full	0.544	0.283	0.372	-	-	-	-	-	-	-	-	full	0.350	0.212	0.264
Lexical Matcher	/1k/1k$_{GQ}$	0.830	0.279	0.418	-	-	-	-	/5k/5k$_{GQ}$	0.415	0.410	0.412	/1k/1k$_{GQ}$	0.528	0.135	0.215
KALM				0.86				-				-				-
CycleNER	/2k/1k$_{GQ}$	0.814	0.837	0.825	/3.4k/1k$_{GQ}$	0.322	0.355	0.338	/116k/5k$_{GQ}$	0.552	0.798	0.653	/12.5k/1k$_{GQ}$	0.337	0.541	0.415
	/14k/14k$_{SQ}$	0.750	0.632	0.686	/3.4k/3.4k$_{SQ}$	0.339	0.360	0.349	/116k/116k$_{SQ}$	0.540	0.709	0.613	/12.5k/12.5k$_{SQ}$	0.332	0.443	0.380

Table 6: Comparison of the performance of the different approaches. The first column in each dataset sub-table represents the dataset used to train the respective models. Refer to Table 3 for the datasets used by CycleNER and Lexical Matcher. For the supervised BERT baseline, we report the size of the parallel data.

much smaller, and in some cases *SQ* sequences allow the model to have better performance, as is the case for WNUT.

On the BC2GM dataset, the gap between BERT and CycleNER increases. This is due to the fact that the T5 model does not contain sufficient pre-trained knowledge, and hence cannot correctly recognize entity boundaries for training S2E and E2S. Often surface forms (e.g. *"HKE6"*) of named entities in BC2GM are tokenized into many subword units, due to the fact that such tokens are not present in the pretraining corpus of T5, resulting in much poorer performance. In this case, we note that since BERT performs token level classification, although far from the state of the art performance, the model can better learn to predict the correct NER class for each token.

Finally, the performance of NeuralHMM and Lexical Matcher is quite poor, which emphasizes the difficulty of conducting this task in an unsupervised fashion. For many of the datasets, e.g. WNUT, OntoNotes, NeuralHMM performs extremely poorly, hence, the results are omitted from the table. Liu et al. [15] report that their unsupervised KALM model can achieve a score of F1=0.86 for CoNLL, while our CycleNER achieves a comparable score of F1=0.83. Considering that CycleNER does not leverage large knowledge bases, its performance is still competitive using only a small entity sample.

Event Location

s : *The festival, in the town's riverside park, was*

Product

held to promote black pearl wax apples...

q : *festival <sep> Event <sep> riverside park <sep> Location <sep> black pearl wax apples <sep> Product*

Figure 5: OntoNotes data and CycleNER output.

8 ANALYSIS OF CYCLENER'S BEHAVIOR

Unseen Entities. When trained on large data (e.g. OntoNotes), CycleNER learns to extract new patterns of *named entities* and *new entity classes* that are not annotated in the original ground-truth.

Figure 5 shows the output of CycleNER, which predicts that *"festival"* is an `event`, *"riverside park"* as a `location`, and *"black pearl wax apples"* as a `product`. Despite the fact that these words were not annotated in the ground-truth, CycleNER determines that they are entities based on the context in which they appear, and on similar entities appearing in the ground-truth. This shows that unsupervised NER and *seq2seq* offers several advantages that can circumvent annotation qualities present in an NER dataset.

Error Analysis. The most frequent errors in CycleNER are related to span detection. This is especially the case when the input sentence contains no entities. In such scenarios, S2E attempts to generate entity sequences with at least one entity.

To better understand this type of behavior, we exclude from the test set all sentences that contain no entities and re-calculate the performance metrics. This results in a 4% absolute increase in terms of F1 for CoNLL, and 8–12% for the other three datasets. Comparatively, BERT as the best performing baseline, sees an increase of 3% for CoNLL and 2–5% for the rest of the datasets.

This is a limitation in our work, as sentences without entities impact both S2E and E2S training. There are two possible directions to address this issue: First, for sentences without entities, S2E can be trained to generate a special token. One drawback of such an approach is to reconstruct back the input sentence in E2S, which would lack the context to map a single special token to all the possible sentences that have no entities. This prerequisite is described in Guo et al. [6] where a bijective mapping between the two modalities is necessary. We aim to solve this issue in future work.

9 CONCLUSION

We presented CycleNER, a novel unsupervised NER approach that uses cycle-consistency training to learn an effective mapping between sentences and entities in two cycles by training S2E and E2S. During the two cycles, the model learns how to reconstruct sentences and entities jointly and interactively. To our best knowledge, this is the first time that cycle-consistency has been applied for NER.

Through extensive experimental evaluations, we confirm the effectiveness of CycleNER on four different datasets, covering different domains (BC2GM, medicine), and data quality (WNUT). To ensure optimal training, we devise a stopping criterion that relies on the loss of E-cycle, which on most of the datasets has a high negative correlation with the actual NER performance. We showed that CycleNER achieves competitive performance w.r.t supervised and baselines like BERT. On different datasets, such as CoNLL, CycleNER reaches 72.7% performance of current state of the art. Since CycleNER only requires sentences and entity examples, it has advantages on recognizing emerging and difficult entities, compared to existing unsupervised baselines relying on external knowledege. In practice, CycleNER can be easily applied to develop an initial NER model and scale up the model by providing more entity examples.

REFERENCES

[1] Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018. Adversarial training for multi-context joint entity and relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2830–2836.

[2] Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the WNUT2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*. 140–147.

[3] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial intelligence* 165, 1 (2005), 91–134. Publisher: Elsevier.

[4] Jiafeng Guo, Gu Xu, Xueqi Cheng, and Hang Li. 2009. Named entity recognition in query. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 267–274.

[5] Qipeng Guo, Zhijing Jin, Xipeng Qiu, Weinan Zhang, David Wipf, and Zheng Zhang. 2020. CycleGT: Unsupervised Graph-to-Text and Text-to-Graph Generation via Cycle Training. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*. 77–88.

[6] Qipeng Guo, Zhijing Jin, Ziyu Wang, Xipeng Qiu, Weinan Zhang, Jun Zhu, Zheng Zhang, and Wipf David. 2021. Fork or fail: Cycle-consistent training with many-to-one mappings. In *Proceedings of International Conference on Artificial Intelligence and Statistics*. PMLR, 1828–1836.

[7] Cong Duy Vu Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. 2018. Iterative Back-Translation for Neural Machine Translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*. Association for Computational Linguistics, 18–24.

[8] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologie*. 4171–4186.

[9] Kai Labusch, Preußischer Kulturbesitz, Clemens Neudecker, and David Zellhöfer. 2019. BERT for Named Entity Recognition in Contemporary and Historical German. In *Proceedings of the 15th Conference on Natural Language Processing, Erlangen, Germany*. 8–11.

[10] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologie*. 260–270.

[11] Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised Machine Translation Using Monolingual Corpora Only. In *Proceedings of International Conference on Learning Representations*.

[12] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

[13] Xiaoya Li, Xiaofei Sun, Yuxian Meng, Junjun Liang, Fei Wu, and Jiwei Li. 2020. Dice Loss for Data-imbalanced NLP Tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 465–476.

[14] Chen Liang, Yue Yu, Haoming Jiang, Siawpeng Er, Ruijia Wang, Tuo Zhao, and Chao Zhang. 2020. Cycle-sum: cycle-consistent adversarial lstm networks for unsupervised video summarization. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1054–1064.

[15] Angli Liu, Jingfei Du, and Veselin Stoyanov. 2019. Knowledge-Augmented Language Model and its Application to Unsupervised Named-Entity Recognition. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologie*. 1142–1150.

[16] Xuezhe Ma and Eduard Hovy. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. 1064–1074.

[17] Tao Meng, Anjie Fang, Oleg Rokhlenko, and Shervin Malmasi. 2021. GEMNET: Effective Gated Gazetteer Representations for Recognizing Complex Entities in Low-context Input. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

[18] Muhammad Tasnim Mohiuddin and Shafiq Joty. 2019. Revisiting Adversarial Autoencoder for Unsupervised Word Translation with Cycle Consistency and Improved Training. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 3857–3867.

[19] Mariana L. Neves and Ulf Leser. 2014. A survey on annotation tools for the biomedical literature. *Briefings Bioinform.* 15, 2 (2014), 327–340.

[20] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, Alessandro Moschitti, Bo Pang, and Walter Daelemans (Eds.). ACL, 1532–1543.

[21] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language*. Association for Computational Linguistics, 2227–2237.

[22] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 21 (2020), 1–67.

[23] Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologie*. ACL, 142–147.

[24] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. 86–96.

[25] Larry Smith, Lorraine K Tanabe, Rie Johnson nee Ando, Cheng-Ju Kuo, I-Fang Chung, Chun-Nan Hsu, Yu-Shi Lin, Roman Klinger, Christoph M Friedrich, Kuzman Ganchev, et al. 2008. Overview of BioCreative II gene mention recognition. *Genome biology* 9, 2 (2008), 1–19.

[26] Jana Straková, Milan Straka, and Jan Hajic. 2019. Neural Architectures for Nested NER through Linearization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 5326–5331.

[27] Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: language-independent named entity recognition. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologie*. Association for Computational Linguistics, 142–147.

[28] Ke M Tran, Yonatan Bisk, Ashish Vaswani, Daniel Marcu, and Kevin Knight. 2016. Unsupervised Neural Hidden Markov Models. In *Proceedings of the Workshop on Structured Prediction for NLP*. 63–71.

[29] Asahi Ushio and Jose Camacho-Collados. 2021. T-NER: An All-Round Python Library for Transformer-based Named Entity Recognition. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*. 53–62.

[30] Leon Weber, Jannes Münchmeyer, Tim Rocktäschel, Maryam Habibi, and Ulf Leser. 2020. HUNER: improving biomedical NER with pretraining. *Bioinform.* 36, 1 (2020), 295–302.

[31] Weischedel, Ralph, Palmer, Martha, Marcus, Mitchell, Hovy, Eduard, Pradhan, Sameer, Ramshaw, Lance, Xue, Nianwen, Taylor, Ann, Kaufman, Jeff, Franchini, Michelle, El-Bachouti, Mohammed, Belvin, Robert, and Houston, Ann. 2021. OntoNotes Release 5.0.

[32] Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. 6442–6454.

[33] Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. 6442–6454.

[34] Shaodian Zhang and Noémie Elhadad. 2013. Unsupervised biomedical named entity recognition: Experiments with clinical and biological texts. *Journal of biomedical informatics* 46, 6 (2013), 1088–1098. Publisher: Elsevier.

[35] Xiangyang Zhou, Daxiang Dong, Hua Wu, Shiqi Zhao, Dianhai Yu, Hao Tian, Xuan Liu, and Rui Yan. 2016. Multi-view Response Selection for Human-Computer Conversation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. The Association for Computational Linguistics, 372–381.

[36] Huiming Zhu, Chunhui He, Yang Fang, and Weidong Xiao. 2020. Fine Grained Named Entity Recognition via Seq2seq Framework. *IEEE Access* 8 (2020), 53953–53961. Publisher: IEEE.

[37] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*. 2223–2232.