Sentence-Level Resampling for Named Entity Recognition

Anonymous ACL submission

Abstract

As a fundamental task in natural language processing, named entity recognition (NER) aims to locate and classify named entities in unstructured text. However, named entities are always the minority among all tokens in the text. This data imbalance problem presents a challenge to machine learning models as their learning objective is usually dominated by the majority of non-entity tokens. To alleviate data imbalance, we propose a set of sentence-level resampling methods where the importance of each training sentence is computed based on its tokens and entities. We study the generalizability of these resampling methods on a wide variety of NER models (CRF, Bi-LSTM, and BERT) across corpora from diverse domains (general, social, and medical texts). Extensive experiments show that the proposed methods improve performance of the evaluated NER models especially on small corpora, frequently outperforming sub-sentence-level resampling, data augmentation, and special loss functions such as focal and Dice loss.

1 Introduction

In natural language processing, named entity recognition (NER) is an important task both on its own and supports numerous downstream tasks such as entity linking and question answering. NER has an inherent data imbalance problem: named entities of interest are almost always the minority among irrelevant (*Other* type) tokens in a text corpus. Table 1 shows the prevalent imbalanced nature of NER corpora from multiple domains. As shown in Table 1, *entity tokens* (tokens associated with any named entity) account for 3.9-16.6% of all tokens in any of these corpora. Within entity tokens, the most frequent entity type may cover 2-200 times more tokens than the least frequent entity type. At the sentence level, 23-85% sentences contain at least one entity, suggesting that 15-77% sentences contain no entity at all.

Data imbalance is even more severe in real-world bespoke NER tasks, which directly motivated this work. For example, given full-text articles from a medical subfield, domain experts may wish to recognize only those concepts related to specific aspects of the subfield (e.g., symptoms and medicine related to a specific disease). Compared to all tokens in the full text, extremely few tokens are annotated with any entity type. Because domain experts have limited availability, annotated corpus are usually small in such tasks. As a result, some rare entity types may have less than 10 tokens across the corpus. Such severe data imbalance and scarcity makes many NER models suffer.

Data imbalance in NER challenges machine learning-based models because their learning objective is dominated by entities of the majority type (*Other*), causing the model to be reluctant to predict the types of interest. Various techniques have been studied to tackle this challenge. Active learning was applied to collect a more balanced dataset at annotation time (Tomanek and Hahn, 2009). Special loss functions including focal loss (Lin et al., 2017) and Dice loss (Li et al., 2019) are proposed to deal with data imbalance. Data augmentation was shown to be effective by enriching entity-bearing sentences through methods like segment shuffling and mention replacement (Dai and Adel, 2020).

The classical method for alleviating data imbalance is resampling (upsampling the minority class or downsampling the majority class) and its close relative, cost-sensitive learning (assigning larger weight to the minority class or smaller weight to the majority class in the learning objective) (He and Garcia, 2009). A natural question is: *Can we apply resampling to address the data imbalance problem in NER?* It turns out that unlike classification tasks, applying resampling to sequence tagging tasks like NER is not straightforward. Recent work attempted sub-sentence-level resampling – dropping tokens from the majority class either

Domain	NER Corpus	# of Tokens	# of Entity Types	% of Least vs. Most Freq. Type Entity Tokens	% of All Entity Tokens	# of Sent.	% of Sent. w/ Entities
Social	WNUT (Derczynski et al., 2017)	59,570	6	0.43 vs. 1.59	5.03	3,394	36.18
General	GMB subset (Bos et al., 2017)	66,161	8	0.03 vs. 4.00	15.03	2,999	85.40
Medical	AnEM (Ohta et al., 2012)	71,697	11	0.03 vs. 1.08	3.91	2,815	35.38
Medical	CADEC (Karimi et al., 2015)	121,307	5	0.21 vs. 6.65	15.76	5,719	58.86
General	CoNLL (Sang and De Meulder, 2003)	204,567	4	2.26 vs. 5.46	16.64	14,986	74.28
Medical	n2c2 ADE (Henry et al., 2020)	813,277	9	0.19 vs. 2.34	10.89	65,293	22.73
General	OntoNotes (Ralph et al., 2013)	2,200,865	18	0.01 vs. 2.59	10.89	115,812	50.11

Table 1: Imbalance ratio statistics in NER corpora from different domains. ('Sent.' = Sentences; 'Freq.' = Frequent)

at random (Akkasi, 2018) or using heuristic rules (Akkasi et al., 2018; Akkasi and Varoglu, 2019; Grancharova et al., 2020). These methods were shown to perform well with shallow NER models – conditional random fields with local n-gram and word shape features. However, sub-sentence-level resampling inevitably destroy the structure of complete sentences and distort the contextual information around entities of interest. Complete sentences are essential for state-of-the-art NER models based on contextual word representations, e.g., deep Transformers (Devlin et al., 2018). As shown in our experiments, incomplete sentences generated by sub-sentence-level resampling often hurt the performance of deep NER models.

In this paper, we propose sentence-level resampling methods for NER, an under-explored problem in this area. As sentences are the natural units of data in NER, sentence-level resampling leaves the contextual information intact in a natural sentence needed by deep models like Transformers. Since a sentence may contain a mixture of entities whose types have different levels of rareness, traditional resampling method for imbalanced classification (e.g., inverse probability resampling) cannot be applied. Instead, we develop a set of methods for computing integer-valued importance score for a sentence based on its entity composition, and resample the sentence accordingly. Experiments show that our methods can improve performance of a variety of NER models and are especially effective on tasks with small annotated corpora, which is often seen in real-world bespoke NER tasks.

2 Related Work

2.1 Learning from Imbalanced Data

Class imbalance is a long-standing problem in machine learning tasks, posing challenges to researchers and practitioners in many domains (King and Zeng, 2001; Lu and Jain, 2003; He and Garcia, 2009; Moreo et al., 2016). Classes in real-world data often have highly skewed distribution, leading to substantial gaps between majority and minority classes. While the positive (minority) class is often of interest, the lack of positive examples makes classifiers conservative, *i.e.*, they incline to predict all example as the negative (majority) class. This often results in a low recall of the positive class. Because only a small number of examples are predicted as positive, precision of the positive class tends to be high or unstable. Such a low-recall, high-precision pattern often hurts the F1-score, the standard metric that emphasizes a balanced precision and recall (Juba and Le, 2019). This performance pattern is observed not only in classification tasks, but also in NER tasks where named entity tokens are the minority compared to non-entity tokens (Mao et al., 2007; Kuperus et al., 2013).

Researchers have proposed various techniques for imbalanced learning, including resampling and cost-sensitive learning (He and Ma, 2013). Both aim to re-balance the representation of different classes in the loss function, such that the classifier is less conservative in making positive predictions. In principle, by equating per-instance resampling frequency with per-instance cost, resampling can be implemented as cost-sensitive learning. However, resampling can be applied to models that do not support cost-sensitive learning, making it conveniently applicable to all models.

2.2 Resampling in Sequence Tagging Tasks

Resampling (and cost-sensitive learning) can be conveniently used in classification and regression tasks where a model makes pointwise predictions (a single categorical or scalar value). Each example has a clearly defined sampling rate (or cost) according to its class label. However, in sequence tagging tasks like NER (more broadly, structured prediction tasks (BakIr et al., 2007; Smith, 2011)), a model predicts multiple values for a sequence (or structured output). For sequence learning algorithms such as linear-chain conditional random fields, while the learning objective is formulated at the sequence level, the evaluation metrics are defined at the entity span level. This makes it non-trivial to determine the sampling rate (or cost) for a sequence that contains tokens from both majority and minority entity types. Simply resampling entities by stripping surrounding context is problematic as sequence tagging algorithms depend on context to make predictions. Recent works proposed to randomly or heuristically drop tokens from sentences to re-balance NER data, which had success using conditional random fields and shallow n-gram features (Akkasi, 2018; Akkasi and Varoglu, 2019; Grancharova et al., 2020). However, these methods distort the syntactic and semantic structure of complete sentences, which may generate low-quality data for models that are capable of capturing long-distance linguistic dependencies (e.g. BERT) and hurt performance of those models. In this work, we focus on resampling strategies that leaves sentences intact.

2.3 Loss Functions for Imbalanced Data

Recent literature proposed special loss functions for tackling data imbalance, including focal loss (Lin et al., 2017) and Dice loss (Li et al., 2019). They increase the cost of 'hard positives' where the correct label has low predicted probability and decrease the cost of 'easy negatives' where the correct label has high predicted probability. However, these loss functions do not fully address data imbalance in NER. First, the formulation does not always emphasize the loss of minority-class tokens – majority-class tokens can also be hard to classify, and minority-class tokens can also be easy to classify. Second, these loss functions only work on token-wise prediction outputs. They cannot work on sequence-level outputs generated by conditional random fields, which is commonly used in NER. Our resampling methods can be seen as estimating sentence-level losses with explicit emphasis on sentences containing minority-class tokens.

3 Resampling Strategy Design

For a sequence tagging task like NER, resampling cannot be as simple as what it is in classification and regression tasks, in which data points can be individually replicated, discarded, or synthesized. In NER, named entities cannot be resampled out of context. The surrounding context of named entities – albeit tokens from the irrelevant *Other* type – should be considered as well. Resampling named entities with context is a double-edged sword: preserving context will help NER models, but too much context increases the amount of non-entity tokens and aggravates the data imbalance problem. The goal of sentence-level resampling is to find the balance between too little and too much context accompanying named entities in complete sentences.

3.1 Sentence Importance Factors in NER

Intuitively, sentences that are worth resampling are those that are more *important* towards constructing a balanced NER dataset. We start by proposing factors that influence the importance of a sentence in resampling. These factors share the theoretical foundation of retrieval functions in information retrieval (Fang et al., 2004). A retrieval function evaluates the utility of a document with respect to the query terms it contains. By direct analogy, a sentence importance score measures the utility of a sentence respect to the entity tokens it contains.

Count of entity tokens. Regardless of entity types, a sentence containing more entity tokens is more important than a sentence filled with non-entity tokens. This factor mirrors *term frequency* in retrieval functions (Salton and Buckley, 1988).

Rareness of entity type. The general idea of resampling for minority classes says that the rarer an entity type is, the more times we should resample sentences containing this type of entity. This factor mirrors *inverse document frequency* in retrieval functions (Salton and Buckley, 1988).

Density of tokens labeled as any entity. Including too much context can aggravate the imbalance problem. While the absolute count of entity tokens matters, the density of entity tokens in a sentence (number of entity tokens compared to the length of a sentence) should also be concerned. The higher the density, the more important a sentence. This factor mirrors *document length normalization* in retrieval functions (Singhal et al., 1996).

Diminishing marginal utility. If one sentence contains twice as many tokens with a specific entity type as the other sentence with the same length, does that mean the first sentence is twice as important as the second? In reality, an entity may contain numerous tokens, or a sentence may include multiple entities of the same type. Twice as many tokens from the same entity type may not offer twice as much information (for the same reason why too many tokens from the *Other* type is not helpful). Therefore, as the number of tokens from the same entity type increases, they generate diminishing marginal utility to a sentence. This factor mirrors *diminishing marginal gain of repeated query terms* in retrieval functions (Fang et al., 2004).

3.2 Resampling Functions

Based on the above importance factors, we design a suite of functions $f_s \in \mathbb{Z}^+$ to determine the number of times a sentence s should be resampled in a NER training set. These functions incorporate progressively more factors discussed previously.

In a given corpus, let us denote the set of all entity types except for the majority type *Other* as T. Let $c(t, s)$ be the count of tokens with entity type $t \in T$ in sentence s. We define the resampling function with respect to the smoothed count (**sC**) of all entity tokens as

$$f_s^{\text{sC}} = 1 + \sum_{t \in T} c(t, s) \ . \qquad (1)$$

Here, $\sum_{t \in T} c(t, s)$ is the total number of entity tokens in sentence s. '+1' is to avoid removing entity-less sentences from the training set, in reminiscence of add-one smoothing in empirical probability estimates. It guarantees that all training sentences are resampled as least once. This smoothing-like process maintains consistency between training and test sets. If the training set contains entity-less sentences, it is highly likely that the test set will contain entity-less sentences as well.

The next function incorporates entity rareness factor. The rareness r_t of an entity type $t \in T$ is measured as the self-information of the event that any token carries this type:

$$r_t = -\log_2 \frac{\sum_{s \in S} c(t, s)}{N} \ ,$$

where S is the set of all sentences in the training set, and therefore $\sum_{s \in S} c(t, s)$ is the total number of tokens with entity type t. N is number of all tokens (including *Other* tokens) in the training set. By introducing rareness of an entity type we propose another function called smoothed resampling incorporating count and rareness (**sCR**):

$$f_s^{\text{sCR}} = 1 + \left\lceil \sqrt{\sum_{t \in T} r_t \cdot c(t, s)} \right\rceil \ . \qquad (2)$$

Ceiling function $\lceil \cdot \rceil$ ensures $f_s^{\text{sCR}} \in \mathbb{Z}^+$. Square root is to slow down the increase of f_s^{sCR} when an entity type t is extremely rare (when r_t is large).

According to the density factor in the previous section, the length of sentence s plays a role in determining the density of entity tokens within a sentence. Let l_s be the length of sentence s measured in number of tokens. We define the following function called the smoothed resampling incorporating count, rareness, and density (**sCRD**):

$$f_s^{\text{sCRD}} = 1 + \left\lceil \frac{\sum_{t \in T} r_t \cdot c(t, s)}{\sqrt{l_s}} \right\rceil \ . \qquad (3)$$

We use $\sqrt{l_s}$ instead of l_s because to slow down the decrease of f_s^{sCRD} when a sentence is too long.

Lastly, we incorporate the diminishing marginal utility factor and propose a function called the normalized and smoothed resampling incorporating count, rareness, and density (**nsCRD**):

$$f_s^{\text{nsCRD}} = 1 + \left\lceil \frac{\sum_{t \in T} r_t \cdot \sqrt{c(t, s)}}{\sqrt{l_s}} \right\rceil \ . \qquad (4)$$

Here, $\sqrt{c(t, s)}$ applies a sublinearly increasing function on $c(t, s)$ to implement the diminishing marginal utility when a sentence contains many tokens with the same type.

In summary, we proposed four functions for determining resampling frequencies for each sentence, representing four resampling methods.

4 Experimental Evaluation

Resampling should be a domain- and model-agnostic strategy in tackling data imbalance. Therefore, the goal of our experiments is to evaluate if the proposed resampling methods are effective in an extensive array of NER corpora and base models. Towards this goal, we apply the four resampling methods (together with baseline methods) on three representative NER models (each has two variants), and evaluate the resulting models on four corpora from diverse domains.

4.1 Evaluation Metric

We use span-level strict-match macro-averaged F1 score as our main evaluation metric. *Other* is not viewed as an entity type. Macro-averaged metrics emphasize a balanced treatment of all entity types, which align with our main goal. See Appendix C for micro-averaged and per-entity-type results.

4.2 Compared Methods

We compare the following baseline methods for dealing with data imbalance in NER.

Original corpus: training data untreated.

Balanced undersampling: We implement the algorithm proposed in (Akkasi et al., 2018) as a representative of sub-sentence-level resampling.

Data augmentation: The data augmentation techniques that includes *all* transformations as proposed in (Dai and Adel, 2020).

Focal loss(Lin et al., 2017), **Dice loss** (Li et al., 2019): We apply these loss functions on token-wise predictions made by a softmax output layer. Note that they are not applicable to sequence-level predictions made by a CRF output layer.

sC, sCR, sCRD, nsCRD: the four resampling methods proposed in this work.

4.3 NER Corpora

We select four corpora from different domains. The first three are of small scale, representing bespoke NER tasks in practice where entity types are task-specific and annotation efforts are limited.

AnEM (Ohta et al., 2012): The Anatomical Entity Mention (AnEM) corpus consists of 500 documents selected randomly from citation abstracts and full-text papers concerning both health and pathological anatomy. With only 3.91% entity tokens and 35.38% sentences having any entity, this is a very imbalanced corpus in Table 1.

WNUT (Derczynski et al., 2017): This is a social domain corpus released in the 2017 Workshop on Noisy User-generated Text (W-NUT). It contains noisy user-generated texts found in social media, online review, crowdsourcing, web forums, clinical records, and language learner essays. This is another very imbalanced corpus in Table 1.

GMB subset (Bos et al., 2017; Kaggle, 2018): The Groningen Meaning Bank (GMB) corpus consists of public domain English texts with corresponding syntactic and semantic representations. The GMB subset is extracted from the larger GMB 2.0.0 corpus which is built specially for NER.

To test the generalizability of our methods, we also include a standard NER benchmark dataset.

CoNLL (Sang and De Meulder, 2003): The CoNLL-2003 English news NER corpus.

For AnEM, WNUT, and CoNLL, we use their pre-existing training/test split. For GMB subset, we use 3:1 training/test split.

4.4 Base NER Models and Variants

To comprehensively evaluate the combinations of our upstream resampling strategies with many downstream sequence tagging models, we select the following models:

Shallow Model. We construct shallow NER models that use pretrained word embeddings as per-word feature vectors. We consider two variants: one using a **softmax** output layer making token-wise predictions; the other using a **CRF** (conditional random fields) output layer making sequence-level predictions. Considering domains of the corpora, we select embeddings trained on biomedical literature (Huang et al., 2016), tweets (Glove-27B-twitter-27B),[1] and Wikipedia+news (Glove-6B),[2] for AnEM, WNUT, and datasets from general domain (GMB subset and CoNLL), respectively. All are 50-dimensional embeddings. CrfSuite[3] is applied with default hyperparameters.

Bi-LSTM (Bidirectional Long Short-Term Memory). LSTM is a special recurrent neural network architecture in which the vanishing gradient problem can be effectively mitigated. Bi-LSTM consists of two LSTMs taking inputs in both forward and backward directions. Even though more recent models (e.g., GPT-2, BERT) are shown to outperform Bi-LSTM, it is still regarded as one of the most prevalent tools for solving sequence tagging problems. We implement two variants of Bi-LSTM: one with a **softmax** output layer making token-wise predictions; the other with a **CRF** decoding layer[4], to ensure the local consistency of output tags. Different from the default hyperparameters, batch size and number of epochs are set to 32 and 20, respectively. Embeddings are used in the same way as in the shallow models above.

BERT (Bidirectional Encoder Representations from Transformers). BERT is widely regarded as the most significant improvement in natural language processing. Its outstanding capability of learning contextualized word representations makes it the representative of advanced NER model in this work. Again, we implement two variants of BERT: one with a **softmax** output layer making token-wise predictions; the other with a **CRF** decoding layer[5]. Default hyperparameters are used. More implementation details are in Appendix A.

[1] http://nlp.stanford.edu/data/glove.twitter.27B.zip
[2] http://nlp.stanford.edu/data/glove.6B.zip
[3] https://github.com/scrapinghub/python-crfsuite
[4] https://github.com/guillaumegenthial/sequence_tagging
[5] https://github.com/kyzhouhzau/BERT-NER

	Shallow model		Bi-LSTM		BERT	
	Softmax	CRF	Softmax	CRF	Softmax	CRF
Original corpus	27.36	52.39	41.76	45.21	58.60	57.68
Balanced undersampling	24.17	52.06	21.22	26.19	62.59	63.03
Data augmentation	22.78	53.20	40.73	41.05	61.34	63.38
Focal loss	27.65	-	40.73	-	58.39	-
Dice loss	25.87	-	2.31	-	47.36	-
sC	30.39	52.13	44.69	47.55	62.86	**65.47**
sCR	**30.94**	53.38	**48.37**	**48.90**	**65.22**	62.41
sCRD	29.98	50.67	45.69	44.31	61.70	60.50
nsCRD	30.57	**53.41**	39.54	46.10	64.63	62.59

Table 2: Macro F1 scores on AnEM. The three NER models using either softmax or CRF output layer are reported. '-' means that focal loss and Dice loss do not apply to CRF outputs. In each column, the highest F1-score is shown in **boldface** and the second highest is shown in underline.

	Shallow model		Bi-LSTM		BERT	
	Softmax	CRF	Softmax	CRF	Softmax	CRF
Original corpus	14.56	4.31	18.27	18.46	37.67	39.98
Balanced undersampling	**16.94**	4.45	15.81	16.68	33.73	30.86
Data augmentation	11.05	4.49	20.10	10.06	35.03	36.08
Focal loss	14.25	-	17.73	-	39.20	-
Dice loss	15.74	-	15.26	-	31.62	-
sC	16.20	4.58	16.90	19.21	37.16	**49.44**
sCR	15.94	4.39	**21.52**	21.40	**44.60**	45.06
sCRD	15.95	**4.94**	17.15	19.80	43.82	42.34
nsCRD	16.06	4.62	18.31	**23.71**	41.65	41.71

Table 3: Macro-averaged F1-scores on the WNUT corpus. See the caption of Table 2 above for details.

	Shallow model		Bi-LSTM		BERT	
	Softmax	CRF	Softmax	CRF	Softmax	CRF
Original corpus	28.71	41.42	45.73	44.91	50.20	54.32
Balanced undersampling	29.56	40.52	41.73	42.70	**57.01**	**56.89**
Data augmentation	27.53	41.24	**47.63**	**49.76**	54.78	55.25
Focal loss	28.47	-	41.25	-	53.66	-
Dice loss	**33.42**	-	40.16	-	52.10	-
sC	29.17	41.34	44.52	48.39	54.52	54.58
sCR	29.85	40.39	45.85	46.91	52.96	54.33
sCRD	30.99	41.54	44.60	48.07	52.72	55.12
nsCRD	29.37	**41.76**	46.16	45.69	55.14	54.60

Table 4: Macro-averaged F1-scores on the GMB subset corpus. See the caption of Table 2 above for details.

	Shallow model		Bi-LSTM		BERT	
	Softmax	CRF	Softmax	CRF	Softmax	CRF
Original corpus	42.66	**67.71**	75.78	78.63	88.20	88.45
Balanced undersampling	42.55	66.77	66.36	76.54	86.91	86.75
Data augmentation	42.73	67.06	77.70	77.31	87.98	88.37
Focal loss	43.47	-	77.24	-	88.44	-
Dice loss	**48.58**	-	72.80	-	88.82	-
sC	42.20	66.84	76.82	75.03	**88.94**	88.81
sCR	42.22	66.63	77.72	74.13	88.36	**89.11**
sCRD	42.98	66.97	78.06	77.63	88.30	88.61
nsCRD	41.98	67.04	**78.80**	**78.87**	87.58	88.03

Table 5: Macro-averaged F1-scores on the CoNLL corpus. See the caption of Table 2 above for details.

4.5 Results and Discussion

Macro-averaged F1-scores of different methods applied to four corpora and three base NER models are reported in Tables 2-5.

Our goal is *not* to compete with state-of-the-art methods on these corpora. Instead, we aim to present an interesting and under-explored problem (sentence-level resampling for NER) and a set of simple yet promising methods. In principle, our proposed resampling methods are model-agnostic and can provide an additional performance boost for a variety of NER models. We observe the following trends in Tables 2-5.

Overall performance of our resampling methods: Across Tables 2-5, our methods (sC, sCR, sCRD, nsCRD) generally performed well, achieving the highest or second highest F1-scores in almost every column (except for the condition 'Shallow model, Softmax' on CoNLL). Although no specific method consistently outperforms others in every condition, it is clear that sentence-level resampling is overall a promising approach to tackling the data imbalance problem in NER. The best resampling method depends on the specific base model, output layer, and corpus used. Just as the best hyperparameter values have to be empirically determined, so could be the most suitable resampling method. Fortunately, all our resampling methods are simple and straightforward, which allows for convenient experimentation.

Shallow vs. deep models: We observe a clear trend that shallow models using word embedding as features and softmax/CRF as the output layer underperform deep models such as Bi-LSTM and BERT. We view this as a sanity check. Bi-LSTMs and BERT can learn word representations that account for long-distance dependencies, and BERT should be even more powerful with contextual word representations pretrained on massive texts.

Softmax vs. CRF output layer: Using the same base model, CRF output layer often (but not always) outperforms softmax output layer. The performance gap is larger on shallow models and small corpora (AnEM, WNUT, GMB) than on deep models and large corpus (CoNLL). Indeed, Bi-LSTM and BERT are capable of learning word representations that account for long-distance word dependencies, reducing the benefit of tag dependencies offered by a CRF layer. Similar observations was made by previous work (Devlin et al., 2018). An exception is the combination (WNUT, Shallow model), where the CRF layer suffered from severe overfitting caused by noisy text and extremely imbalanced data distribution in WNUT corpus.

Small vs. large corpus: On small corpora (AnEM, WNUT, GMB subset), our resampling methods usually outperform the original corpus baseline by a big margin. These benefit becomes less salient on large corpus (CoNLL). This implies that our methods are especially effective when the corpus is small and annotations are few. As corpus size gets large, even rare entity types are covered by many examples and therefore sufficiently trained.

Sub-sentence resampling and data augmentation: Sub-sentence resampling (balanced undersampling) has large variance in its performance. In some cases it gives the highest gain (GMB subset, BERT model), and in other cases it performs worse than just using the original corpus (all corpora, Bi-LSTM models). It suggests that sub-sentence resampling is highly sensitive to the corpus and model choice. Data augmentation also shows high variance in its performance. It gives the highest gain on (GMB subset, Bi-LSTM model), and performs worse than the original corpus on (WNUT, BERT model). Sentences generated by data augmentation generally have correct syntax but garbled semantics (e.g., one entity is replaced by another same-type, out-of-context entity). The nonsensical sentences may confuse NER models. In contrast, whole-sentence resampling methods give more stable improvements over the original corpus baseline largely because they preserve the naturalness of resampled sentences.

Focal loss and Dice loss: These loss functions are applicable only on pointwise predictions made by the softmax output layer. A major trend is that their performance tend to be unreliable across scenarios. We attribute this behavior to the difficulty in optimizing these losses. For shallow models, we optimize them by feeding gradients of either loss function (see Appendix B for their derivation) into a L-BFGS optimizer (Liu and Nocedal, 1989) in Scikit-Learn. As shown in the (Shallow model, Softmax) column of GMB subset and CoNLL corpora, the two loss functions (especially the Dice loss) performed well. For deep models (Bi-LSTM and BERT), we rely on TensorFlow's automatic differentiation and Adam gradient descent optimizer (Kingma and Ba, 2014) because manually deriving gradients for deep models is infeasible. The two loss functions sometimes give poor performance.

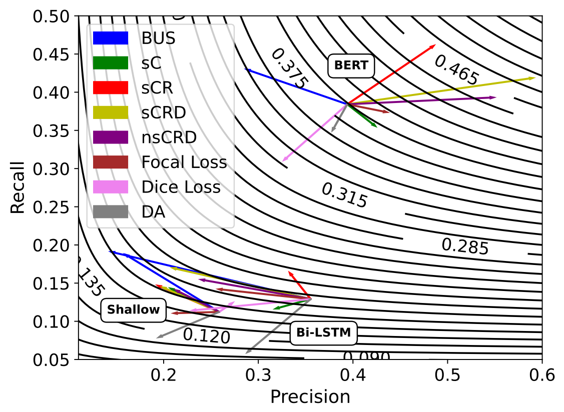

Figure 1: Displacement vectors of F1-scores in precision-recall plots for WNUT corpus. BUS: balanced undersampling. DA: data augmentation. All models use softmax output layer. The downward curves are contours of F1-scores in the precision-recall space. Each NER model (Shallow, Bi-LSTM, BERT) is associated with a cluster of vectors sharing the same starting point in the space, which represents the performance on the original corpus.

The Bi-LSTM model with Dice loss failed completely on AnEM (F1-score: 2.31). A possible explanation is that Dice loss is non-convex and it may be difficult for first-order optimizers in current deep learning toolkits (e.g. Adam in TensorFlow) to find high-quality local minima than second-order methods like L-BFGS.

Precision and recall: To illustrate performance changes in terms of precision and recall, Figure 1 visualizes the changes before and after resampling as displacement vectors in precision-recall plots with F1-score contour lines. Some arrows are pointing to the upper right corner of the plots, indicating the associated methods improve F1-score by improving both precision and recall. Other arrows point to the upper left, indicating the associated methods increase recall at the sacrifice of precision. In this case, most of our methods improve macro-averaged precision and recall of the BERT model on WNUT. See Appendix C.2 for more details.

4.6 Effect on Training Corpus Size

Table 6 shows the effect of training corpus size as a result of resampling or data augmentation. These factors are the average across four corpora.

The balanced undersampling method drops tokens from sentences, and therefore reduces training corpus size. Data augmentation method doubles the corpus size as many sentences are paraphrased into multiple versions. Our proposed methods increases corpus size by a larger factor because sentences that contain rare entity types are resampled many times. Although increased training corpus size leads to increased training time, note that our methods are especially suitable for scenarios where the annotated corpus is small and hence the training time is still relatively short.

Methods	Size increase factor
Original corpus	1.00
Balanced undersampling	0.32
Data augmentation	2.00
sC	3.80
sCR	4.60
sCRD	3.91
nsCRD	2.82

Table 6: Effect of data resampling/augmentation methods on training corpus size. The factors are averaged across four evaluated corpora.

5 Conclusion

Our proposed sentence resampling methods generalize well across diverse NER corpora and models. They enjoy the following advantages:

Model-agnostic: Since resampling only manipulates datasets and not models, the proposed methods can be directly applied to any NER model, requiring no knowledge of its functioning or any change to it. Resampling is also more convenient than cost-sensitive learning as the latter still requires changing the model training process.

Domain-agnostic: Compared with data preprocessing methods such as data augmentation, sentence-level resampling methods are simple and do not require domain- or language-specific manipulations such as synonym replacement, saving practitioners from excessive data engineering.

There are multiple avenues for future work. First, further theoretical and empirical research can explore more effective resampling functions that deliver consistently better performance across corpora and base models. Second, more corpora and models can be examined under these resampling strategies. Third, the variance of performance in different scenarios may potentially relate to characteristics of specific corpora. Future research may seek for corpora-level statistics that can assist practitioners in the process of selecting the appropriate resampling method(s).

References

A Akkasi and E Varoglu. 2019. Improvement of chemical named entity recognition through sentence-based random under-sampling and classifier combination. *Journal of AI and Data Mining*, 7(2):311–319.

Abbas Akkasi. 2018. Sentence-based undersampling for named entity recognition using genetic algorithm. *Iran Journal of Computer Science*, 1(3):165–174.

Abbas Akkasi, Ekrem Varoğlu, and Nazife Dimililer. 2018. Balanced undersampling: a novel sentence-based undersampling method to improve recognition of named entities in chemical and biomedical text. *Applied Intelligence*, 48(8):1965–1978.

Gökhan BakIr, Thomas Hofmann, Bernhard Schölkopf, Alexander J Smola, and Ben Taskar. 2007. *Predicting structured data*. MIT press.

Johan Bos, Valerio Basile, Kilian Evang, Noortje Venhuizen, and Johannes Bjerva. 2017. The groningen meaning bank. In Nancy Ide and James Pustejovsky, editors, *Handbook of Linguistic Annotation*, volume 2, pages 463–496. Springer.

Xiang Dai and Heike Adel. 2020. An analysis of simple data augmentation for named entity recognition. *arXiv preprint arXiv:2010.11683*.

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*, pages 140–147.

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*.

Hui Fang, Tao Tao, and ChengXiang Zhai. 2004. A formal study of information retrieval heuristics. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 49–56.

Mila Grancharova, Hanna Berg, and Hercules Dalianis. 2020. Improving named entity recognition and classification in class imbalanced swedish electronic patient records through resampling. In *Eighth Swedish Language Technology Conference (SLTC)*. Förlag Göteborgs Universitet.

Haibo He and Edwardo A Garcia. 2009. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284.

Haibo He and Yunqian Ma. 2013. Imbalanced learning: foundations, algorithms, and applications.

Sam Henry, Kevin Buchan, Michele Filannino, Amber Stubbs, and Ozlem Uzuner. 2020. 2018 n2c2 shared task on adverse drug events and medication extraction in electronic health records. *Journal of the American Medical Informatics Association*, 27(1):3–12.

Jian Huang, Keyang Xu, and VG Vinod Vydiswaran. 2016. Analyzing multiple medical corpora using word embedding. In *2016 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 527–533. IEEE.

Brendan Juba and Hai S Le. 2019. Precision-recall versus accuracy and the role of large data sets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4039–4048.

Dataset Kaggle. 2018. Annotated GMB corpus. https://www.kaggle.com/shoumikgoswami/annotated-gmb-corpus. Accessed: 2021-05-10.

Sarvnaz Karimi, Alejandro Metke-Jimenez, Madonna Kemp, and Chen Wang. 2015. Cadec: A corpus of adverse drug event annotations. *Journal of biomedical informatics*, 55:73–81.

Gary King and Langche Zeng. 2001. Logistic regression in rare events data. *Political analysis*, 9(2):137–163.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jasper Kuperus, Cor J Veenman, and Maurice van Keulen. 2013. Increasing ner recall with minimal precision loss. In *2013 European Intelligence and Security Informatics Conference*, pages 106–111. IEEE.

Xiaoya Li, Xiaofei Sun, Yuxian Meng, Junjun Liang, Fei Wu, and Jiwei Li. 2019. Dice loss for data-imbalanced nlp tasks. *arXiv preprint arXiv:1911.02855*.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

Dong C Liu and Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528.

Xiaoguang Lu and Anil K Jain. 2003. Resampling for face recognition. In *International Conference on Audio-and Video-based Biometric Person Authentication*, pages 869–877. Springer.

Xinnian Mao, Wei Xu, Yuan Dong, Saike He, and Haila Wang. 2007. Using non-local features to improve named entity recognition recall. In *Proceedings of the 21st Pacific Asia Conference on Language, Information and Computation*, pages 303–310.

Alejandro Moreo, Andrea Esuli, and Fabrizio Sebastiani. 2016. Distributional random oversampling for imbalanced text classification. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 805–808.

Tomoko Ohta, Sampo Pyysalo, Jun'ichi Tsujii, and Sophia Ananiadou. 2012. Open-domain anatomical entity mention detection. In *Proceedings of the workshop on detecting structure in scholarly discourse*, pages 27–36.

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. 2013. Ontonotes release 5.0. https://doi.org/10.35111/xmhb-2b84.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.

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 the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Amit Singhal, Gerard Salton, Mandar Mitra, and Chris Buckley. 1996. Document length normalization. *Information Processing & Management*, 32(5):619–633.

Noah A Smith. 2011. Linguistic structure prediction. *Synthesis lectures on human language technologies*, 4(2):1–274.

Katrin Tomanek and Udo Hahn. 2009. Reducing class imbalance during active learning for named entity annotation. In *Proceedings of the fifth international conference on Knowledge capture*, pages 105–112.

A Implementation Details

A.1 Software and Hardware Environment

All the deep learning models are implemented in Tensorflow 1.12.0 environment.

Softmax regression (or multinomial logistic regression) model is from Scikit-Learn package in version 0.23.2. The CRF model is implemented by the package sklearn-crfsuite in version of 0.3.6.

Data resampling and CRF training/evaluation were performed on 2.60 GHz Intel CPUs and 8GB RAM. Bi-LSTM and BERT training/evaluation were performed on GPUs (GeForce GTX1080 8GB and Tesla V100 16GB).

A.2 Hyperparameters for Machine Learning Models

For shallow models and BERT, all hyperparameters are set by default. For details of them, please see documents of sklearn, crfsuite and BERT-NER.

For Bi-LSTM, we adjust a few of parameters as there are some drawbacks of the default settings: 20 is not a commonly used number for batch size, and loss of Bi-LSTM model fails to converge under some circumstances. So we set them to 32 and 20, instead of default values 20 and 15. Other hyperparameters are applied according to default settings.

For the fairness in the comparison, we do not alter any hyperparameters while switching resampling methods and loss functions without changing dataset and models. We believe that it is totally appropriate in the process of comparing, despite that better performance of specific methods may be obtained after tuning hyperparameters, which beyond the scope of this exploring research.

A.3 Hyperparameters for Loss Functions

There are two hyperparameters in the focal loss and Dice Loss, determining converging speed and smooth degree. For focal loss, we set γ to 2, as what authors of (Lin et al., 2017) recommend. According to (Li et al., 2019), it is appropriate to set γ of Dice loss to 1 for the purpose of smoothing. In our implementation of loss function in shallow model, this setting is found effective. However, while using it in deep learning model, its effectiveness cannot be ensured. Hence, we adopt another setting of $\gamma = 10^{-5}$ in the tensor computing and obtain better results compared with those obtained with a larger γ.

B Derivation of Loss Function Gradients for Softmax Regression

When using the shallow model with softmax output layer and focal/Dice loss functions, we optimize the model parameters by the quasi-Newton method L-BFGS provided by Python Scikit-Learn. This approach requires us to provide the gradients of current model parameters. Below we show our derivation of these gradients.

Notations and Preliminaries. Scalar values are denoted by non-bold, lowercase letters such as x. Row vectors are denoted by bold, lowercase letters such as $\mathbf{x}$. Matrices are denoted by bold, uppercase letters such as $\mathbf{X}$.

Softmax regression has the following components:

- Feature vector: $\mathbf{x} \in \mathbb{R}^m, \mathbf{x} = [x_1, \cdots, x_i, \cdots, x_m]$.
- Label vector: $\mathbf{y} \in \{0,1\}^k, \mathbf{y} = [y_1, \cdots, y_j, \cdots, y_k]$. If the ground truth is the c-th class, $1 \le c \le k$, then $y_c = 1$, and $y_j = 0$ if $j \ne c$.
- Weight vector for the j-th class: $\mathbf{w}_j \in \mathbb{R}^m, \mathbf{w}_j = [w_{j1}, \cdots, w_{ji}, \cdots, w_{jm}]$.
- Weight matrix $\mathbf{W} \in \mathbb{R}^{m\times k}$, $\mathbf{W} = [\mathbf{w}_1^\top, \cdots, \mathbf{w}_j^\top, \cdots, \mathbf{w}_k^\top]$. "$\top$" is the transpose operation. $\mathbf{w}^\top$ is the transpose of $\mathbf{w}$, which is a column vector.
- Bias for the the j-th class: $b_j \in \mathbb{R}$.
- Predicted probability vector: $\mathbf{p} \in [0,1]^k, \mathbf{p} = [p_1, \cdots, p_j, \cdots, p_k]$.

$$p_j = \Pr(y_j = 1|\mathbf{x}) \tag{5}$$

$$= \frac{\exp(\langle \mathbf{w}_j, \mathbf{x}\rangle + b_j)}{\sum_{j'=1}^{k} \exp(\langle \mathbf{w}_{j'}, \mathbf{x}\rangle + b_{j'})} \tag{6}$$

$\langle \mathbf{w}, \mathbf{x}\rangle$ is the inner product of vector $\mathbf{w}$ and vector $\mathbf{x}$.

One can verify that the partial derivative of p_c with respect to w_{ji}, the weight of the j-th class, the i-th dimension, is the following:

$$\frac{\partial p_c}{\partial w_{ji}} = [\mathbf{1}\{j = c\} - p_j]\, p_c x_i \tag{7}$$

B.1 Focal Loss Gradient

Suppose the ground truth is the c-th class for a given example $\mathbf{x}$.

$$L_{FL}(\mathbf{x}, \mathbf{y}) = -(1 - p_c)^\gamma \log p_c \tag{8}$$

$$\begin{aligned}
&\frac{\partial L_{FL}(\mathbf{x}, \mathbf{y})}{\partial w_{ji}} && (9)\\
=&-\frac{\partial}{\partial p_c}[(1-p_c)^\gamma \log p_c] \cdot \frac{\partial p_c}{\partial w_{ji}} && (10)\\
=&-\left[-\gamma(1-p_c)^{\gamma-1}\log p_c + \frac{(1-p_c)^\gamma}{p_c}\right] \cdot \frac{\partial p_c}{\partial w_{ji}} && (11)\\
=&-\left[-\gamma(1-p_c)^{\gamma-1}\log p_c + \frac{(1-p_c)^\gamma}{p_c}\right]\\
&\cdot [\mathbf{1}\{j=c\} - p_j] p_c x_i && (12)\\
=&-[-\gamma p_c(1-p_c)^{\gamma-1}\log p_c + (1-p_c)^\gamma]\\
&\cdot [\mathbf{1}\{j=c\} - p_j] x_i && (13)\\
=&a_c[p_j - \mathbf{1}\{j=c\}] x_i && (14)
\end{aligned}$$

Here we set

$$a_c = -\gamma p_c(1-p_c)^{\gamma-1}\log p_c + (1-p_c)^\gamma \tag{15}$$

to reduce notational clutter. a_c has nothing to do with i or j; it only has to do with c, the index of the ground truth label for the training example $\mathbf{x}$. When $\gamma = 0$, $a_c = 1$. When $\gamma > 0$, a_c decreases when p_c increases from 0 to 1. This means the gradient for an easy example (when p_c is close to 1) have a smaller magnitude than the gradient for a hard example (when p_c is close to 0).

Generalizing the scalar gradient in Equation (14) to matrix gradient, we have

$$\frac{\partial L_{FL}(\mathbf{x}, \mathbf{y})}{\partial \mathbf{W}} = a_c \cdot \mathbf{x}^\top (\mathbf{p} - \mathbf{y})\,. \tag{16}$$

The shape of $a_c \cdot \mathbf{x}^\top(\mathbf{p} - \mathbf{y})$ is $m \times k$, the same shape as $\mathbf{W}$.

An important note is that here a_c is specific to that single example $\mathbf{x}$, which has ground truth label $y_c = 1$. If we have n different training examples $\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(n)}$, then every example will have a different a_c value: $a_c^{(1)}, \cdots, a_c^{(n)}$. Let's create a diagonal matrix $\mathbf{A}_c \in \mathbb{R}^{n\times n}$, $\mathbf{A}_c = diag(a_c^{(1)}, \cdots, a_c^{(n)})$.

If we have n training examples, then the feature matrix $\mathbf{X} \in \mathbb{R}^{n\times m}$, the label matrix $\mathbf{Y} \in \{0,1\}^{n\times k}$, and the predicted probability matrix $\mathbf{P} \in [0,1]^{n\times k}$. We have:

$$\frac{\partial L_{FL}(\mathbf{X}, \mathbf{Y})}{\partial \mathbf{W}} = \mathbf{X}^\top \mathbf{A}_c (\mathbf{P} - \mathbf{Y})\,. \tag{17}$$

The shape of $\mathbf{X}^\top \mathbf{A}_c(\mathbf{P} - \mathbf{Y})$ is $m \times k$, the same as $\mathbf{W}$.

B.2 Dice Loss Gradient

Dice loss computes per-class F-1 scores. Suppose the ground truth is the c-th class for a given example $\mathbf{x}$.

$$\begin{aligned}
&L_{DL}(\mathbf{x}, \mathbf{y}) && (18)\\
=&\sum_{j'=1}^{k}\left[1 - \mathbf{1}\{c = j'\}\frac{\gamma + 2p_c}{\gamma + p_c^2 + 1}\right.\\
&\left.+1 - \mathbf{1}\{c \neq j'\}\frac{\gamma}{\gamma + p_{j'}^2}\right] && (19)\\
=&1 - \frac{\gamma + 2p_c}{\gamma + p_c^2 + 1} + \sum_{j'\neq c}\left[1 - \frac{\gamma}{\gamma + p_{j'}^2}\right] && (20)\\
=&k - \frac{\gamma + 2p_c}{\gamma + p_c^2 + 1} - \sum_{j'\neq c}\frac{\gamma}{\gamma + p_{j'}^2} && (21)
\end{aligned}$$

Take gradient with respect to w_{ji}, the weight of the j-th class, the i-th dimension.

$$\begin{aligned}
&\frac{\partial L_{DL}(\mathbf{x}, \mathbf{y})}{\partial w_{ji}} && (22)\\
=&-\frac{\partial}{\partial p_c}\left[\frac{\gamma + 2p_c}{\gamma + p_c^2 + 1}\right] \cdot \frac{\partial p_c}{\partial w_{ji}}\\
&-\sum_{j'\neq c}\frac{\partial}{\partial p_{j'}}\left[\frac{\gamma}{\gamma + p_{j'}^2}\right] \cdot \frac{\partial p_{j'}}{\partial w_{ji}} && (23)\\
=&-\frac{2(\gamma + p_c^2 + 1) - (\gamma + 2p_c)2p_c}{(\gamma + p_c^2 + 1)^2} \cdot \frac{\partial p_c}{\partial w_{ji}}\\
&-\sum_{j'\neq c}\frac{-\gamma \cdot 2p_{j'}}{(\gamma + p_{j'}^2)^2} \cdot \frac{\partial p_{j'}}{\partial w_{ji}} && (24)\\
=&-\frac{2(\gamma + p_c^2 + 1) - (\gamma + 2p_c)2p_c}{(\gamma + p_c^2 + 1)^2}\\
&\cdot [\mathbf{1}\{j = c\} - p_j] p_c x_i\\
&-\sum_{j'\neq c}\frac{-\gamma \cdot 2p_{j'}}{(\gamma + p_{j'}^2)^2} \cdot [\mathbf{1}\{j = j'\} - p_j] p_{j'} x_i && (25)\\
=&-\frac{2(1 - p_c)(1 + \gamma + p_c)p_c}{(\gamma + p_c^2 + 1)^2} \cdot [\mathbf{1}\{j = c\} - p_j] x_i && (26)
\end{aligned}$$

$$+\sum_{j'\neq c}\frac{\gamma\cdot 2p_{j'}^2}{(\gamma+p_{j'}^2)^2}\cdot[\mathbf{1}\{j=j'\}-p_j]x_i \quad (27)$$

$$=a_c[p_j-\mathbf{1}\{j=c\}]x_i-\sum_{j'\neq c}b_{j'}[p_j-\mathbf{1}\{j=j'\}]x_i \quad (28)$$

Here we set

$$a_c=\frac{2(1-p_c)(1+\gamma+p_c)p_c}{(\gamma+p_c^2+1)^2} \quad (29)$$

$$b_{j'}=\frac{\gamma\cdot 2p_{j'}^2}{(\gamma+p_{j'}^2)^2} \quad (30)$$

a_c depends on the ground truth label of example $\mathbf{x}$. $b_{j'}$ depends on the current predicted probabilities for $\mathbf{x}$.

Generalizing the scalar gradient in Equation (28) to matrix gradient, we have

$$\frac{\partial L_{DL(\mathbf{x},\mathbf{y})}}{\partial \mathbf{W}} \quad (31)$$

$$=\mathbf{x}^\top a_c(\mathbf{p}-\mathbf{y})$$

$$-\mathbf{x}^\top\left[\underbrace{\cdots,\sum_{j'\neq c}b_{j'}[p_j-\mathbf{1}\{j=j'\}],\cdots}_{j=1,\cdots,k}\right] \quad (32)$$

$$=\mathbf{x}^\top a_c(\mathbf{p}-\mathbf{y})-\mathbf{x}^\top\mathbf{v} \quad (33)$$

$\mathbf{v}$ is a vector specific to the example $\mathbf{x}$.

If we have n training examples, then the feature matrix $\mathbf{X}\in\mathbb{R}^{n\times m}$, the label matrix $\mathbf{Y}\in\{0,1\}^{n\times k}$, and the predicted probability matrix $\mathbf{P}\in[0,1]^{n\times k}$. We have:

$$\frac{\partial L_{DL}(\mathbf{X},\mathbf{Y})}{\partial \mathbf{W}}=\mathbf{X}^\top\mathbf{A}_c(\mathbf{P}-\mathbf{Y})-\mathbf{X}^\top\mathbf{V} \quad (34)$$

where $\mathbf{V}$ has shape $n\times k$, and the l-th row in matrix $\mathbf{V}$ is a k dimensional vector computed in the same manner as Equation (32) with respect to the l-th training example, $1\leq l\leq n$.

C Additional Performance Analysis

C.1 Micro-averaged Metrics

In the main paper we reported macro-averaged F1 scores for each dataset. To provide a more complete comparison of performance changes, here we report micro-averaged F1 scores in Tables 7-10. Micro-averaged metrics lump together all named entities without distinguishing their types, and therefore the majority types have more influence on these metrics than minority types. Overall, the trend is consistent with the macro-averaged metrics. Sentence-level resampling methods tend to deliver more stable gains and generally outperform baseline methods.

C.2 Per-Entity-Type Metrics

To further examine the impacts of our method on entity types, we also report per-entity-type precision, recall, and F1 scores for each dataset in Tables 11-14. We compare the performance of using the original corpus and a representative of our methods (sCR). Red up-arrows (↑) means sCR obtains better precision/recall/F1 score compared to using the original corpus.

Here we observe that at the level of entity types, either precision and recall simultaneously improve or drop, or precision improves at a slight cost of recall. It is rare that recall improves at the cost of precision (only the GPE type in Table 13). This pattern indicates that the BERT-CRF model trained on the original corpus has many 'false negatives' (tagging entity tokens as "other"). In other words, the model is extremely reluctant to predict non-other entity types. Our sentence-level resampling methods encourage the model to correctly assign entity types to more tokens. Another trend is that improvements on small corpora (AnEM, WNUT, GMB subset) are more salient than on large corpus (CoNLL). Note that sentence resampling does not necessarily favor minority entity types as all entity types are very rare already, compared to the *Other* tokens (see the last column of Tables 11-14, "Token %").

	Shallow model		Bi-LSTM		BERT	
	Softmax	CRF	Softmax	CRF	Softmax	CRF
Original corpus	33.96	56.74	55.94	58.81	67.02	66.69
Balanced undersampling	31.12	56.03	24.95	30.90	69.62	69.38
Data augmentation	32.70	56.80	53.12	56.19	68.09	**71.63**
Focal loss	28.89	-	53.12	-	68.24	-
Dice loss	**35.62**	-	8.62	-	64.50	-
sC	35.00	55.75	57.03	59.24	70.90	70.55
sCR	35.25	55.89	**57.12**	**60.37**	71.76	69.71
sCRD	34.04	55.65	55.57	57.02	70.22	69.57
nsCRD	35.28	**56.87**	54.08	58.52	**71.08**	69.82

Table 7: Micro-averaged F1-scores on AnEM. The three NER models using either softmax or CRF output layer are reported. '-' means that focal loss and Dice loss do not apply to CRF outputs. In each column, the highest F1-score is shown in **boldface** and the second highest is shown in underline.

	Shallow model		Bi-LSTM		BERT	
	Softmax	CRF	Softmax	CRF	Softmax	CRF
Original corpus	21.74	5.28	27.89	29.32	61.31	60.70
Balanced undersampling	22.77	5.69	22.51	24.64	53.70	51.66
Data augmentation	17.33	5.47	28.73	14.37	60.00	61.86
Focal loss	21.62	-	30.23	-	62.18	-
Dice loss	**24.01**	-	29.53	-	57.14	-
sC	22.74	5.62	25.49	28.55	63.54	64.62
sCR	23.25	**6.48**	**31.38**	27.76	65.67	**65.99**
sCRD	23.09	6.01	27.74	32.69	**66.02**	64.68
nsCRD	23.17	5.68	28.83	**34.29**	62.00	63.41

Table 8: Micro-averaged F1-scores on the WNUT corpus. See the caption of Table 7 above for details.

	Shallow model		Bi-LSTM		BERT	
	Softmax	CRF	Softmax	CRF	Softmax	CRF
Original corpus	50.60	66.67	**75.92**	76.87	80.43	81.68
Balanced undersampling	50.94	66.18	70.91	73.86	77.87	78.41
Data augmentation	49.09	66.13	74.75	76.17	81.48	81.86
Focal loss	49.84	-	72.44	-	81.61	-
Dice loss	**57.70**	-	70.46	-	80.02	-
sC	50.68	66.32	75.85	**77.11**	81.26	81.16
sCR	51.68	66.09	74.76	76.29	80.45	**82.12**
sCRD	51.12	66.58	72.43	74.21	80.71	81.09
nsCRD	50.76	**66.87**	73.38	74.62	**81.69**	81.54

Table 9: Micro-averaged F1-scores on the GMB subset corpus. See the caption of Table 7 above for details.

	Shallow model		Bi-LSTM		BERT	
	Softmax	CRF	Softmax	CRF	Softmax	CRF
Original corpus	42.47	**71.11**	79.84	81.03	92.08	92.23
Balanced undersampling	43.22	70.43	71.46	76.75	90.97	90.78
Data augmentation	42.37	70.84	81.87	81.40	91.96	92.33
Focal loss	43.78	-	81.53	-	92.15	-
Dice loss	**48.77**	-	77.51	-	91.94	-
sC	43.42	70.86	81.10	78.70	**92.53**	92.48
sCR	42.77	70.57	82.26	78.18	91.88	**92.59**
sCRD	44.48	70.84	82.29	81.36	91.93	92.07
nsCRD	43.24	70.67	**83.05**	**83.42**	91.37	91.87

Table 10: Micro-averaged F1-scores on the CoNLL corpus. See the caption of Table 7 above for details.

	Original corpus			sCR			
Entity Type	P	R	F	P	R	F	Token %
Developing anatomical structure	86.96	90.91	88.89	87.50↑	95.45↑	91.30↑	0.03
Immaterial anatomical entity	24.14	23.33	23.73	40.00↑	40.00↑	40.00↑	0.06
Anatomical system	52.17	85.17	64.86	41.38↓	85.71↑	55.81↓	0.08
Organism subdivision	42.86	54.00	47.79	64.44↑	58.00↑	61.05↑	0.16
Cellular component	26.37	27.27	26.81	41.79↑	31.82↑	36.13↑	0.21
Tissue	35.62	52.34	42.46	46.97↑	52.54↑	49.60↑	0.27
Organism substance	57.58	77.87	66.21	74.62↑	80.83↑	77.60↑	0.30
Organ	78.95	75.47	77.17	78.47↓	71.07↓	74.59↓	0.35
Pathological formation	62.18	65.54	63.82	72.73↑	59.46↓	65.43↑	0.51
Multi-tissue structure	50.17	60.08	54.68	56.45↑	57.61↓	57.02↑	0.86
Cell	77.02	79.07	78.03	79.79↑	76.33↓	78.02↓	1.08
Macro-avg	54.00	62.89	57.68	62.19↑	64.44↑	62.41↑	-

Table 11: Per-entity-type precision (P), recall (R), and F1 scores (F) on AnEM corpus using BERT-CRF model.

	Original corpus			sCR			
Entity Type	P	R	F	P	R	F	Token %
Corporation	33.33	66.67	44.44	20.00↓	33.33↓	43↓	0.43
Creative work	0	0	0	100.00↑	20.00↑	33.33↑	0.55
Product	33.33	33.33	33.33	25.00↓	16.67↓	20.00↑	0.55
Group	30.77	12.90	18.18	47.37↑	29.03↑	36.00↑	0.66
Location	70.00	72.41	71.18	91.30↑	72.41	80.76↑	1.27
Person	71.11	74.42	72.72	76.19↑	74.42	75.29↑	1.59
Macro-avg	39.76	43.29	39.98	59.98↑	40.98↓	45.06↑	-

Table 12: Per-entity-type precision (P), recall (R), and F1 scores (F) on WNUT corpus using BERT-CRF model.

	Original corpus			sCR			
Entity Type	P	R	F	P	R	F	Token %
NAT	0	0	0	0	0	0	0.03
ART	12.00	10.34	11.11	0↓	0↓	0↓	0.07
EVE	26.67	36.36	30.77	35.71↑	45.45↑	40.00↑	0.11
GPE	54.38	52.49	53.41	54.20↓	52.99↑	53.59↑	1.58
TIM	76.69	88.73	88.27	77.22↑	89.71↑	83.00↑	2.40
ORG	74.39	77.06	76.72	77.43↑	75.76↓	76.59↓	3.31
PER	86.87	87.31	87.09	88.56↑	89.90↑	89.22↑	3.56
GEO	93.19	93.19	93.19	92.26↓	92.26↓	92.26↓	4.00
Macro-avg	53.27	55.68	54.32	53.17↓	55.86↑	54.33↑	-

Table 13: Per-entity-type precision (P), recall (R), and F1 scores (F) on GMB subset using BERT-CRF model.

	Original corpus			sCR			
Entity Type	P	R	F	P	R	F	Token %
MISC	77.26	81.76	79.44	82.61↑	82.49↑	82.54↑	2.26
LOC	92.17	92.50	92.33	91.76↓	91.49↓	91.62↓	4.07
ORG	86.16	88.83	87.47	85.92↓	88.59↓	87.23↓	4.92
PER	94.44	94.66	94.55	95.11↑	94.97↑	95.03↑	5.46
Macro-avg	87.51	89.44	88.45	88.85↑	89.38↓	89.11↑	-

Table 14: Per-entity-type F1 scores on CoNLL using BERT-CRF model.