User-in-the-Loop Named Entity Recognition by Counterfactual Learning

Tong Yu
Adobe Research
San Jose, CA, USA
tyu@adobe.com

Junda Wu
New York University
New York City, NY, USA
jw6466@nyu.edu

Ruiyi Zhang
Adobe Research
San Jose, CA, USA
ruizhang@adobe.com

Handong Zhao
Adobe Research
San Jose, CA, USA
hazhao@adobe.com

Shuai Li*
Shanghai Jiao Tong University
Shanghai, China
shuaili8@sjtu.edu.cn

Abstract

Named Entity Recognition (NER) is an important task, to enable a wide range of NLP applications. The state-of-the-art NER models are based on deep learning and require enough labeled data. In practice, the labeled data for NER is usually limited, as providing accurate labels to the sentences is very time consuming. With an NER model trained on limited labeled data, it is desirable to develop an efficient mechanism to collect data labels and improve the model over time. To achieve this, existing works develop active learning approaches. However, these approaches are usually developed for annotators and assume the annotators will provide the exactly correct labels to each sentence selected to label. In this paper, we propose a simple yet effective user-in-the-loop feedback mechanism to enable end users, instead of annotators, to easily provide labels to the system. We identify counterfactual bias of the data collected by this feedback mechanism. To alleviate the bias and achieve more sample-efficient learning, we further develop a counterfactual NER learning framework. We develop an imputation model to estimate the loss in those non-displayed entity classes. By considering both losses on displayed and non-displayed entity classes, we can efficiently alleviate such display bias in the NER model. With extensive experiments, we validate the effectiveness of our feedback mechanism and learning framework.

1 Introduction

In the task of Named Entity Recognition (NER), we extract and classify named entities in text data into pre-defined categories. The state-of-the-art NER models are developed based on deep neural networks, where an adequate amount of labeled data is needed to train a generalize model. However, the labeled NER data is usually limited in practice, since it requires a lot of human effort to provide accurate labels to the sentences. With limited labeled data to train an NER model, it is desirable to develop an efficient mechanism to continually collect data labels and update the model over time.

To collect data and update the NER model, existing works develop active learning approaches [14, 6, 13]. These approaches are usually developed for annotators (*e.g.*, from Amazon Mechanical Turk), who are usually hired to provide correct labels to the sentences or subsequences. By selecting appropriate sentences or subsequences and require the labels from the annotators, we can effectively

*Corresponding author

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Virtual.

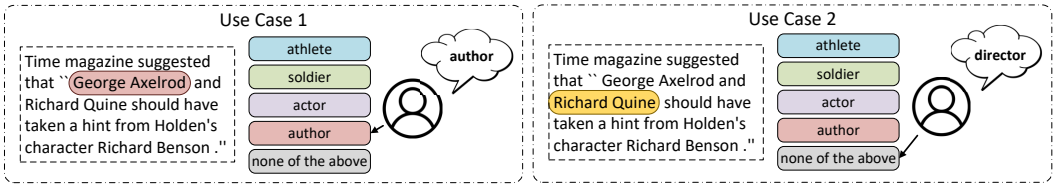

Figure 1: Use cases of our NER system. In a modern NER system, there may be hundreds of entity classes. In use case 1, the user gets a list of predicted top candidate entities and provides the feedback that 'author' is correct. In use case 2, the user cannot find the correct entity within the list, and choose 'none of the above'. Directly training on the data collected in use case 2 may be problematic, due to a selection bias between distributions of displayed and non-displayed entities [20].

improve the accuracy of the NER system. However, requiring labels from annotators may still be limited. In practice, the steaming data of new classes keeps arriving. To train an NER model to classify the data of new classes, we need to continually hire annotators to provide labeled data of new classes, which can be very expensive. Besides, the customers who are actually using the NER system (*i.e.*, end users) may have various background. For example, some end users may care about the named entities from medical domains while some other end users may care about the named entities from travel domains. Therefore, it is desirable to collect the feedback directly from the end users, instead of annotators, to improve the NER model.

In this paper, we propose a simple yet effective user-in-the-loop feedback mechanism to enable end users, instead of annotators, to easily provide labels to the NER system. Each time when the user receives the prediction results from the system, it is designed that the user will receive a list of candidate prediction results. Then, the user can select the correct one from the list of candidates, or provide the feedback that none of the candidates is correct. The use case 1 in Figure 1 shows an example. The user get a list of predicted candidate entities and provides the feedback that 'author' is correct. Based on the user feedback, the NER model updates and improves over time.

Although the above user-in-the-loop feedback mechanism is straightforward, we identify an important challenge when developing algorithms to efficiently learn from the user feedback collected by the above mechanism. The initial NER model may be trained on limited labeled sentences. When the labeled sentences are limited, there are data selection bias which will also lead to the bias of the initial NER model. With this inaccurate initial model, the correct entities are not able to be displayed as the top candidates and receive positive feedback from the user, especially when there are many entities but only limited space is available to display the candidates. Without handling the non-displayed correct entities, the NER model updated over time may also have a strong bias and inaccurate predictions, due to the inconsistency between distributions of labeled and unlabeled samples [20]. As a result, the NER model learning over time is suboptimal and we cannot achieve sample-efficient learning.

To address the above challenge, we further develop ULNER-DR: a user-in-the-loop NER learning framework based on the doubly robust method. First, we introduce an uniform policy to online collect unbiased data and train an imputation model. Second, to alleviate the bias, we develop the counterfactual loss based on the imputation model to guide the model training process. With the imputation model and counterfactual loss, we can effectively alliviate the bias and achieve more sample-efficient learning. Our main contributions are summarized as follows.

- We propose a simple yet effective feedback mechanism to enable end users to easily provide labels to the system. We identify counterfactual bias when developing algorithms to efficiently learn from the user feedback collected by the proposed mechanism.
- We develop a counterfactual NER learning framework to alleviate the bias and achieve more sample-efficient learning.
- With extensive experiments on real-world datasets, we validate the effectiveness of our feedback mechanism and counterfactual NER learning framework.

2 Related Work

Named Entity Recognition on Streaming Data Several works are developed to enable the NER model to incorporate new labeled data and learn. One direction is to develop active learning

mechanism, where the system interactively selects samples to require labels from humans [14, 6, 13]. In [14], various selection strategies are implemented, in which the best strategy matches state-of-the-art performance using only 25% of original data. To achieve more sample-efficient learning, [13] enables the active learning algorithm to query subsequences within sentences, and further propagate their labels to other sentences. Our design is different from active learning for NER [14, 6, 13]. These approaches [14, 6, 13] are usually developed for annotators, and assume that the annotators will provide the exactly correct labels to each sentence or subsequence selected to be annotated. In this paper, we focus on a different setting where the system interactively selects samples to require labels from end users of the NER system. Different from annotators, the end users pays to get the labeling services, and it is unrealistic to assume the end users to label the exactly correct entity class especially when there are many entity classes (*e.g.*, there are 66 entity classes in [4]). Instead, the end user may only provide feedback to some top candidate predictions, as illustrated in Figure 1.

Counterfactual Learning Previous works [9, 16, 20] realized the importance of handling non-displayed events. They regarded non-displayed events as unlabeled instances, and model the CTR prediction as a learning problem with labeled and unlabeled instances, which aims to learning under covariate shift (sample bias corrections). Several counterfactual estimators have been developed. Importance sampling (IS) is a simple way to tackle this issue, but suffers from high variances. Classic variance reduction techniques [2] and additive control variates [11] for IS are useful for counterfactual evaluation and learning. The self-normalized estimator [17] is superior to the vanilla IS estimator and obtains improved performance. Inverse propensity score (IPS) [8] weights each labeled event with the inverse of its propensity score, which is determined by the tendency or the likelihood of the logged data. Doubly robust for counterfactual learning [5] takes advantage of the IPS [8] and direct method [20]: if either one of the two estimators is correct, then the estimation is unbiased, increasing the chances of accurate ratio estimations. Our approach based on doubly robust is a simple and effective way for the user-in-the-loop NER learning problem, showing great improvements.

3 NER Model Architecture

In our user-in-the-loop NER learning setting, the model needs to be frequently retrained while collecting human feedback. To develop a practical solution, we accelerate the computation by following the model architecture in [14]. The NER model mainly consists of three components. First, characters in each word are represented as character-level embeddings $\{\mathbf{x}_{ij}\}_{i=1,j=1}^{M,C}$, where M represents the number of words in a sentence and C represents the number of characters in each word. The character encoder takes the inputs of character-level embeddings and extracts a global feature vector $\{\mathbf{w}_i^c\}_{i=1}^{M}$ of each word. Second, in addition to character-level embeddings, we further include word-level embeddings $\{\mathbf{w}_i^w\}_{i=1}^{M}$ as part of words' representations. The word encoder encodes both $\mathbf{w}_i^c$ and $\mathbf{w}_i^w$ to the final word representations $\{\mathbf{h}_i\}_{i=1}^{M}$. Third, by calculating the sequential information of $\{\mathbf{h}_i\}_{i=1}^{M}$ in the sentence, the entity decoder generates the prediction of entity types of each word. We introduce the details of the model as follows.

Character Encoder Following [14], we adopt the CNNs [10] to encode characters $\{x_{ij}\}_{i=1,j=1}^{M,C}$. We set the size of the receptive field to $k_c = 3$ and conduct depthwise convolution on each channel. The channel size in character encoder is set to $d_c = 50$. The non-linear activation function between convolutional layers is ReLU [12]. The output layer of the character encoder is *max pooling* which outputs the global feature vector $\{\mathbf{w}_i^c\}_{i=1}^{M}$ of the word.

Word Encoder The word embeddings in our model consists of two parts of effect. First part is extracted from the character encoder $\{\mathbf{w}_i^c\}_{i=1}^{M}$. In addition, we calculate extra word embeddings $\{\mathbf{w}_i^w\}_{i=1}^{M}$. The dimension of the word embeddings is $d_w = 300$. Thus, we represent the full word embeddings as the concatenation of these two parts $\mathbf{w}_i^f = [\mathbf{w}_i^w, \mathbf{w}_i^c]$. Similar to the character encoder, we adopt the CNNs to encoder word-level features but without the last layer of *max pooling*. We set the size of the receptive field to $k_w = 5$ and also use ReLU as the activation function between layers. The final word representations are denoted as $\{\mathbf{h}_i\}_{i=1}^{M}$.

Entity Decoder We decode the entity labels using a LSTM model. With the obtained sequence of word representations $\{\mathbf{h}_i\}_{i=1}^{M}$, we recurrently decode each entity label first as a sequence of latent

vectors $\{\mathbf{z}_i\}_{i=1}^{M}$. Then, to predict the entity label of each word, we add an additional linear layer to output the predicted logits before the $\mathrm{Softmax}$ layer. For the i-th word token in the sentence, the entity decoder outputs the predicted probability distribution $\{p_{i,k}\}_{k=1}^{K}$ of the target entity type $\{y_{i,k}\}_{k=1}^{K}$, where K is the number of total entity types and $y_{i,k}$ is a binary variable to indicate whether the entity type k is the correct label class or not.

4 Counterfactual NER Learning Framework

4.1 Motivation: Counterfactual Bias Leads to Sample-Inefficient Learning

When directly training NER models on the data collected by the human-in-the-loop mechanism, potentially there are severe counterfactual bias which leads to sample-inefficient model learning. With the user feedback, the NER system naturally learns over time. However, with limited labeled data initially, there are data selection bias and the initial NER model is biased and inaccurate. With this inaccurate initial model, the correct entities are not able to be displayed as the top candidates and receive positive feedback from the user, especially when there are many entities (*e.g.*, as in [4]) but only limited space is available to display the candidates. Without handling the non-displayed entities, the NER model updated over time may also have a strong bias and inaccurate predictions, due to the inconsistency between distributions of labeled and unlabeled samples [20]. As a result, the NER model learning is not sample efficient and accurate over time.

The above bias is even more severe when the number of entity classes increases over time. In real world, we want to enable the NER system to learn to classify data of new entity classes over time. This can be achieved by the end users providing labels of new (unseen before) classes over time. However, when an entity class is unseen in the initial training data but appear in the sentence to label, this entity is unlikely to be selected as the top candidates and receive the positive feedback from the user. Consider the use case 2 in Figure 1 and assume the 'director' class is unseen when obtaining the current NER model. By the current NER model, 'director' cannot be displayed as the top candidate and receive the positive feedback from the user. As a result, the above bias inevitably happens.

4.2 Doubly Robust User-in-the-Loop NER (ULNER-DR)

To eliminate the bias, not only displayed entities but also non-displayed entities should be considered. To handle bias between distributions of displayed and non-displayed entities, we develop a counterfactual NER learning framework. As in [5, 20], the doubly robust method for counterfactual learning takes advantage of the IPS and DM: if either one of the two estimators is correct, then the estimation is unbiased, increasing the chances of accurate ratio estimations. Besides, the doubly robust method is simple and introduces limited extra computational cost. Therefore, we develop ULNER-DR: a user-in-the-loop NER learning framework based on the doubly robust method.

For each round of interaction, the incoming batch of entity $\{x_t\}_{t=1}^{L}$ is required to be annotated. The NER model will first give the predicted top-K entity types for each entity $\hat{y}_{t,j} = f(x_t)$. Then, the human will give the feedback $y_{t,j}$ on the display types which is regarded as the ground truth for the NER model f. Based on the prediction and the human feedback, we develop the *cross-entropy* loss to train the NER model

$$l_{CE}(\hat{y}_{t,j}, y_{t,j}) = -y_{t,j} \log\left(p(\hat{y}_{t,j})\right). \tag{1}$$

To alleviate the display bias, we introduce an imputation model by a uniform policy which is not a trained policy but a policy randomly sample K types of entities and display to the human. In such process, the policy online collects some data to further train an imputation model σ. Since we assume such imputation model is unbiased, we use it to guide our NER model f to learn from the effect in those non-displayed types. We further involve the prediction $\hat{y}^{\sigma}_{t,j} = \sigma(x_t)$ of the imputation model on both the displayed types and the non-displayed types. Following [20], we develop the loss function

$$l_{DR} = \sum_{j}^{K} \left(l_{CE}(\hat{y}_{t,j}, y_{t,j}) - \omega l_{CE}(\hat{y}_{t,j}, \hat{y}^{\sigma}_{t,j})\right) + \sum_{j}^{\hat{K}} \omega l_{CE}(\hat{y}_{t,j}, \hat{y}^{\sigma}_{t,j}) \tag{2}$$

in our counterfactual NER learning framework.

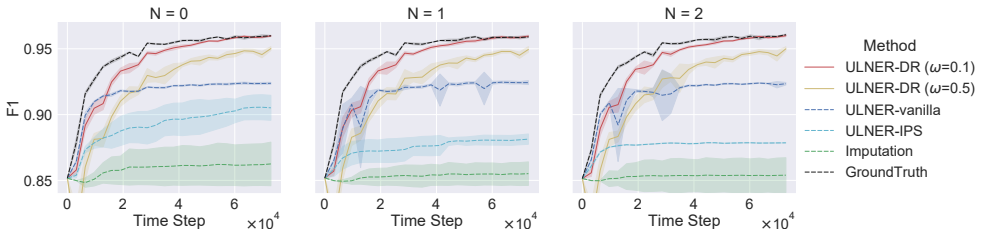

Figure 2: Comparisons between different approaches on CoNLL-2003.

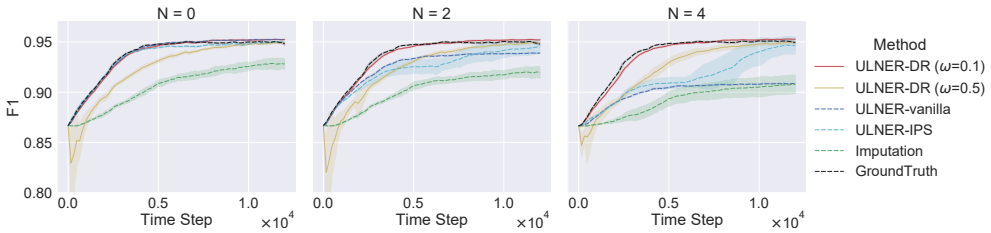

Figure 3: Comparisons between different approaches on GMB.

5 Experiment

Dataset We evaluate different approaches on three datasets: CoNLL-2003 [19], GMB [1], and Few-NERD [4]. CoNLL-2003 [19] includes 4 types of entities and contains about $14K$ training data and $3.4K$ test data. The GMB [1] dataset includes 8 types of entities. This data has about $4.6K$ data, and we split it into $2.5K$ training data and $0.8K$ test data. To study a more challenging and practical setting, we evaluate different approaches on the Few-NERD dataset [4] which contains 66 classes of entities. The Few-NERD dataset has about $188.2K$ training data and $37.6K$ test data.

Evaluation Setup and Metric To evaluate different approaches over time, the training process is split into pretraining and online learning. In the pretraining, we assume only limited labeled data is available. Specifically, we use 1% of the training data to train an initial NER model. Afterwards, the initial NER model interacts with the end user on the rest 99% training data, receives user feedback, and learns online. During the interactions, the NER model expects to receive feedback from the end users. Similar to the evaluations in other interactive applications such as dialog systems and interactive recommenders [3, 7, 15], one challenge is that it is unrealistic to access to user feedback to any possible system actions. Similar to [3, 7, 15, 18], we develop a user simulator to provide user feedback. We use the same model architecture and the same model capacity for the user simulator as for the NER model. We train the simulator on the training data, by the *cross-entropy* loss in Equation 1 with the ground truth entity types as the target $y_{i,k}$. The the user simulator interacts with our NER system as follows. During each interaction, the NER model provides its top-K predicted candidates. Then, the user simulator predicts the ground truth entity class. If the user simulator finds the correct entity class in the candidates, it gives the positive feedback on the found entity class in the candidates. Otherwise, it will provide a feedback indicating that none of the above is correct. Similar to [14], we calculate the F1 score of the models on the test data after each interaction (*i.e.*, time step). The average results over 10 runs with standard errors are reported.

Baseline We report the results of our **ULNER-DR** when $\omega = 0.1$ and 0.5. With a smaller ω, the imputation model has less effect on the method. We further compare **ULNER-DR** to several baselines: (i) **ULNER-vanilla.** This is an variant of ULNER-DR, by only using the imputation model to calculate the loss function without interacting with human simulator. (ii) **ULNER-IPS.** This is also an variant of ULNER-DR, by only considering the loss $l_{CE}(\hat{y}_{t,j}, y_{t,j}) - \omega l_{CE}(\hat{y}_{t,j})$ in Equation 2. (iii) **Imputation.** Without using the trained NER model to interactively collect human feedback, this method only interact with the human based on the uniform policy. The uniform policy simply gives the candidate types by randomly sample K types from all entity types. With the only collected data, we train the imputation model and construct the loss to fine-tune our NER model. (iv) **GroundTruth.** By this approach, the NER model is trained with the ground truth data. That is, during each interaction, it is assumed that the model can always receive the feedback indicating the exactly correct labels. As discussed in Section 1 and Figure 1, this assumption is unrealistic especially,

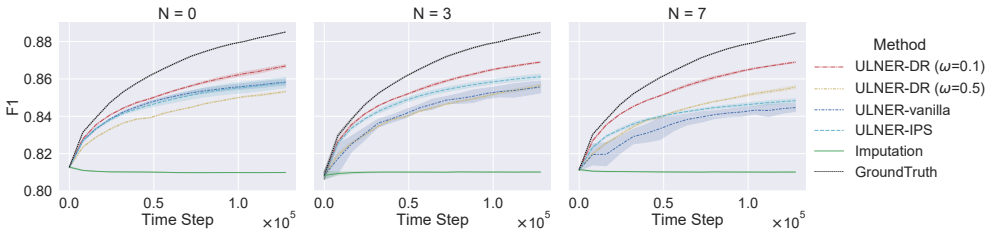

Figure 4: Comparisons between different approaches on Few-NERD.

when there are many entity classes while the space of displaying the entities to the users is limited. Thus, the performance of GroundTruth is expected to be the upper bound of all other approaches. By observing the gap between different approaches and GroundTruth, we can understand the learning sample-efficiency of different approaches. As discussed in Section 3, in the user-in-the-loop NER learning setting, the model needs to be frequently retrained while collecting human feedback. To develop a practical solution, we accelerate the computation by following the model architecture [14] for all the above baselines.

Results of Our Feedback Mechanism We validate our user-in-the loop feedback mechanism. Specifically, we compare our approach ULNER-DR to GroundTruth, which has access to the perfectly correct labels at each time and is the upper bound of ULNER-DR. The results are shown in Figure 2, 3 and 4. Our approach steadily improves over time and achieves very similar performance, compared to GroundTruth. On the CoNLL-2003, ULNER-DR performs only 0.47% lower than GroundTruth, while ULNER-DR has the same performances as GroundTruth on both GMB and Few-NERD, after $12K$ and $128K$ time steps, respectively. Compared to the gaps between GroundTruth and ULNER-DR, the gaps between GroundTruth and Imputation, ULNER-IPS and ULNER-vanilla are much more significant. This validates that ULNER-DR is much more sample efficient than Imputation, ULNER-IPS and ULNER-vanilla.

Results of Alleviating the Counterfactual Bias We validate the improvements by alleviating the bias during the interactions. As discussed in Section 4.1, the bias is higher and learning becomes more challenging when the number of classes is more limited in the training data at the initial time step. To evaluate different approaches in this more challenging setting, we remove N classes in the training data at the initial time step. Specifically, we set $N = \{0, 1, 2\}$ for CoNLL-2003, set $N = \{0, 2, 4\}$ for GMB and set $N = \{0, 3, 7\}$ for Few-NERD. The results are shown in Figure 2, 3 and 4. There are several observations. First, ULNER-DR outperforms ULNER-vanilla, by alleviating the counterfactual bias. On CoNLL-2003, ULNER-DR has achieves 3.90% final improvement over ULNER-vanilla. On Few-NERD, ULNER-DR gains 1.67% improvement over ULNER-vanilla. On GMB, ULNER-DR and ULNER-vanilla have the similar performances. Second, when there are more unseen new classes, our approach's improvement over the baselines is more significant. On CoNLL-2003, when there are 1 and 2 classes unseen, our approach gains 3.82% and 3.95% after $73K$ time steps respectively. On GMB, after $12K$ time steps, the improvement is 1.41% when 2 classes are unseen and the improvement increases to 4.89% when 4 classes are unseen. Third, on the more challenging dataset Few-NERD with more entity classes, our approach's improvements over the baselines are more significant. On Few-NERD, when there are 3 classes unseen, after $128K$ time steps, our approach outperforms the baseline by 1.18% and such improvement increases to 3.01% if 7 classes are unseen in the initial training stage.

6 Conclusion

The state-of-the-art NER models are usually based on deep neural networks, where the model training relies on an adequate amount of labeled data. However, in practice it requires a lot of human effort to obtain rich labeled data. With too limited labeled data to train an accurate NER model, it is desirable to enable the NER model to receive user feedback and learn over time. To achieve this, we develop a simple yet effective feedback mechanism to enable end users to easily provide labels to the NER system. We further discover that learning NER models under the proposed feedback mechanism is non-trivial, by identifying the counterfactual bias. To alleviate the bias and achieve more sample-efficient learning, we develop a counterfactual NER learning framework.

References

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

[2] Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X Charles, D Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. Counterfactual reasoning and learning systems: The example of computational advertising. *JMLR*, 2013.

[3] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. Towards conversational recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 815–824, 2016.

[4] Ning Ding, Guangwei Xu, Yulin Chen, Xiaobin Wang, Xu Han, Pengjun Xie, Hai-Tao Zheng, and Zhiyuan Liu. Few-nerd: A few-shot named entity recognition dataset. *arXiv preprint arXiv:2105.07464*, 2021.

[5] Miroslav Dudík, John Langford, and Lihong Li. Doubly robust policy evaluation and learning. *arXiv preprint arXiv:1103.4601*, 2011.

[6] Meng Fang, Yuan Li, and Trevor Cohn. Learning how to active learn: A deep reinforcement learning approach. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 595–605, 2017.

[7] Xiaoxiao Guo, Hui Wu, Yu Cheng, Steven Rennie, Gerald Tesauro, and Rogério Schmidt Feris. Dialog-based interactive image retrieval. In *NeurIPS*, 2018.

[8] Daniel G Horvitz and Donovan J Thompson. A generalization of sampling without replacement from a finite universe. *Journal of the American statistical Association*, 47(260):663–685, 1952.

[9] Himabindu Lakkaraju, Jon Kleinberg, Jure Leskovec, Jens Ludwig, and Sendhil Mullainathan. The selective labels problem: Evaluating algorithmic predictions in the presence of unobservables. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 275–284, 2017.

[10] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[11] Lihong Li, Rémi Munos, and Csaba Szepesvári. Toward minimax off-policy value estimation. In *AISTATS*. PMLR, 2015.

[12] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[13] Puria Radmard, Yassir Fathullah, and Aldo Lipani. Subsequence based deep active learning for named entity recognition. In *ACL/IJCNLP (1)*, volume 1, pages 4310–4321. Association for Computational Linguistics, 2021.

[14] Yanyao Shen, Hyokun Yun, Zachary C Lipton, Yakov Kronrod, and Animashree Anandkumar. Deep active learning for named entity recognition. In *International Conference on Learning Representations*, 2018.

[15] Weiyan Shi, Kun Qian, Xuewei Wang, and Zhou Yu. How to build user simulators to train rl-based dialog systems. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1990–2000, 2019.

[16] Adith Swaminathan and Thorsten Joachims. Batch learning from logged bandit feedback through counterfactual risk minimization. *The Journal of Machine Learning Research*, 16(1):1731–1755, 2015.

[17] Adith Swaminathan and Thorsten Joachims. The self-normalized estimator for counterfactual learning. *advances in neural information processing systems*, 2015.

[18] Ryuichi Takanobu, Runze Liang, and Minlie Huang. Multi-agent task-oriented dialog policy learning with role-aware reward decomposition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 625–638, 2020.

[19] Erik F. Tjong Kim Sang and Fien De Meulder. 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, 2003.

[20] Bowen Yuan, Jui-Yang Hsia, Meng-Yuan Yang, Hong Zhu, Chih-Yao Chang, Zhenhua Dong, and Chih-Jen Lin. Improving ad click prediction by considering non-displayed events. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 329–338, 2019.