Learning Structured Representations of Entity Names using Active Learning and Weak Supervision

Kun Qian, Poornima Chozhiyath Raman, Lucian Popa, Yunyao Li
IBM Research
qian.kun@ibm.com, {pchozhi, lpopa, yunyaoli}@us.ibm.com

Abstract

Structured representations of entity names are useful for many entity-related tasks such as entity normalization and variant generation. Learning the implicit structured representations of entity names without context and external knowledge is particularly challenging. In this paper, we present a novel learning framework that combines active learning and weak supervision to solve this problem. Our experimental evaluation show that this framework enables the learning of high-quality models from merely a dozen or so labeled examples.

1 Introduction

Entity normalization and variant generation are fundamental for a variety of other tasks such as semantic search and relation extraction (Bhutani et al., 2018; Arasu and Kaushik, 2009). Given an entity name E, the goal of entity normalization is to convert E to a canonical form (e.g., "*Jordan, Michael*" $\rightarrow$ "*Michael Jordan*"), while the goal of entity variant generation is to convert E to a set of different textual representations that refer to the same entity as E (e.g., "*Michael Jordan*" $\rightarrow$ {"*Jordan, Michael*", "*MJ*", "*M. Jordan*", ...}).

Typically, entity normalization and variant generation are done by first performing entity linking (Moro et al., 2014; Zhao et al., 2019; Li et al., 2017), i.e., matching entity names appearing in some context (e.g., free text) to named entities in curated knowledge bases (KBs), then use the canonical form or variations (of the linked entities) residing in the KBs to complete the tasks. Unfortunately, in some scenarios, such as search (Thompson and Dozier, 1997), entity names are not surrounded by context. Furthermore, for specialized domain-specific applications, there may not be a knowledge base to govern the names of the relevant entities. Thus, entity linking is not always applicable. In this paper, we take the view that entity normalization and variant generation can be done without contextual information or external KBs if we understand the internal structures of entity names.

As observed in (Campos et al., 2015; Bhutani et al., 2018; Arasu and Kaushik, 2009; Katiyar and Cardie, 2018; Finkel and Manning, 2009), entity names often have implicit structures that can be exploited to solve entity normalization and variant generation. Table 1 shows how we can manipulate such structured representations of entity names to generate different variations without help from context or external knowledge.

Declarative frameworks are proposed in (Arasu and Kaushik, 2009; Campos et al., 2015) to allow developers to manually specify rules that parse entity names into a structured representation. To avoid such low-level manual effort, (Katiyar and Cardie, 2018; Finkel and Manning, 2009) used fully supervised methods for identifying nested entities embedded in flat named entities. Unfortunately, labeled data are rarely available to leverage these methods in the real-world. To mitigate the need for training data, (Bhutani et al., 2018; Qian et al., 2018) proposed an active learning system, LUSTRE, to semi-automatically learn rules for mapping entity names to their structured representations. By using regex-based extractors and a list of comprehensive dictionaries that capture crucial domain vocabularies, LUSTRE can generate rules that achieve SoTA results. However, for more complex and realistic scenarios, dictionaries may not be available and regex-based extractors alone are not expressive enough. Moreover, as shown in Section 3, LUSTRE cannot handle long entities such as machine logs.

In this paper, we present a framework that learns high-quality BERT-CRF models for parsing entity names into structured representations in low-resource settings, namely, when no labeled data is

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 6376–6383,
November 16–20, 2020.

Mention	Structured Representation	Manipulation	Variations
Michael Jordan	"*Michael*"⟨first⟩ "*Jordan*"⟨last⟩	⟨last⟩,⟨first⟩	*Jordan, Michael*
		createInitial(⟨first⟩) ⟨last⟩	*M Jordan*
		createInitial(⟨first⟩) *createInitial*(⟨last⟩)	*MJ*
General Electric Company	"*General Electric*"⟨name⟩ "*Company*"⟨suffix⟩	*createInitial*(⟨name⟩) *drop*(⟨suffix⟩)	*GE*
		createInitial(⟨name⟩) *abbreviate*(⟨suffix⟩)	*GE Co.*

Table 1: Normalization & variant generation by manipulating structured representation of entity names

available. The proposed framework is essentially an active learning-based approach that learns from human interactions. We believe that comprehensible user interfaces are essential for active learning-based approaches, especially for labeling tasks that require non-trivial human labels (e.g., sequence labels in our approach). Therefore, we developed a system named PARTNER (Qian et al., 2020) that implements this framework. We designed the interface of PARTNER similar to that of LUSTRE, but we also made major modifications so that it is more user friendly. Interested readers can find a video demo of PARTNER at http://ibm.biz/PARTNER. Our main contributions include:

- A hybrid framework combining active learning and weak supervision to effectively learn BERT-CRF-based models with low human effort.
- A full-fledged system, with intuitive UI, that implements the framework.
- Comprehensive experimental results showing that the framework learns high-quality models from merely a dozen or so labeled examples.

Related work. Our problem is related to both flat and nested named entity recognition (NER). However, as discussed in (Finkel and Manning, 2009), NER focuses on identifying the outermost flat entities and completely ignores their internal structured representations. (Katiyar and Cardie, 2018; Ju et al., 2018; Finkel and Manning, 2009; Dinarelli and Rosset, 2012) identify nested entities within some context using fully supervised methods that require large amounts of labeled data, whereas our goal is to learn from very few labels (e.g., < 15) in a contextless fashion. Active learning (Settles, 2009) and weak supervision have been widely adopted for solving many entity-centric problems, such as entity resolution (Kasai et al., 2019; Qian et al., 2019, 2017; Gurajada et al., 2019), NER (Lison et al., 2020; Shen et al., 2018; He and Sun, 2017; Nadeau, 2007), and entity linking (Chen and Ji, 2011). While the power of the combination of the two techniques has been demonstrated in other domains (e.g., computer vision (Brust et al., 2020)), to the best of our knowledge, the two approaches are usually applied in isolation in prior entity-related work.

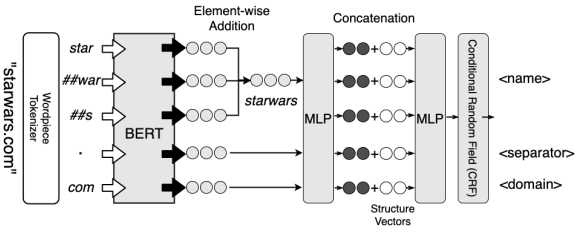


Figure 1: BERT-CRF based model

Recently, data programming approaches (e.g., (Ratner et al., 2017; Safranchik et al., 2020)) use labeling functions/rules to generate weak labels to train machine learning models in low-resource scenarios. Data programming approaches like Snorkel usually assume that labeling functions are manually provided by users, indicating that their target users must have programming skills in order to provide such labeling functions. In contrast, our goal is to minimize both human effort (i.e., minimize labeling requests) and lower human skills (no programming skills are needed).

2 Methodology

Given a set $E = \{\mathcal{E}_1, \ldots, \mathcal{E}_m\}$ of isolated entity mentions (name strings) of a particular type, where $\mathcal{E}_i$ is a sequence $\mathcal{E}_i = (t_i^1, \ldots, t_i^n)$ of tokens. Assume that the input set E of entity names contain a set $C = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ of semantic components (i.e., labels such as ⟨first⟩, ⟨middle⟩ in person names). Our goal is to learn a labeling model $\mathcal{M} : \mathcal{E}_i = (t_i^1, \ldots, t_i^n) \rightarrow (y_1, \ldots, y_n)$, where $y_k \in C$. The labeling model $\mathcal{M}$ is a BERT-CRF based model (see Fig. 1) with several key modifications, which we elaborate next.

Tokenization & vectorization. An input entity name is tokenized with BERT's wordpiece tokenizer, which may result in sub-words for out-vocabulary tokens, e.g., "*starwars*" $\rightarrow$ {"*star*", "*##war*", "*##s*"}. In this case, we combine these sub-words' embeddings (from BERT) into one vector using element-wise addition (see Fig. 1). We then feed the sequence of token embeddings to a

multi-layer perceptron (MLP), the goal of which is to condense the BERT embeddings to smaller embeddings (e.g., 50), so that they are somewhat comparable to the size of the *structure vectors* (to be discussed next), which are crucial for our active learning and weak supervision approach. It is not hard to see that the pre-trained BERT model can be replaced with any other seq2seq models with pre-trained static word embeddings such as BiLSTM + fastText (Bojanowski et al., 2016).

Structure vectors. We predefined a set of boolean predicates where each of them verifies whether or not a token satisfies a specific syntactic pattern. In our experiments, we defined a list of 15 predicates, which can be easily extended, as shown below:

```
hasAllCapsTokens()
hasAllLowerTokens()
hasAllAlphbeticalToken()
hasPunctuationOnly()
isAlphanumToken()
containsNumber()
containsPunctuation()
isFirstLetterCapitalized()
isTwoDigitNumber()
isFourDigitNumber()
isSingleDigitNumber()
isInteger()
isNumericToken()
appearAtBegining()
appearAtEnd()
```

Each token is then converted to a boolean vector using the predefined boolean predicates, and is concatenated with the corresponding condensed token embedding emitted from the first MLP (see Fig 1). Intuitively, condensed token embeddings can capture semantic information and structure vectors can capture structural information.

CRF layer. Each of the concatenated vector are fed to another MLP, which condense them into a vector of size $|C|$ (i.e., the number of label classes). Finally, the final CRF layer uses viterbi algorithm to find out the most likely sequence of labels using the emission vectors (i.e., embeddings from the last MLP layer) and learned transition matrix.

2.1 Weak Supervision with Structure Vectors

Recall that each token is associated with a binary structure vector that carries its "structure" information. Consider the following company names:

- **"Apple Inc." = {"*Apple*", "*Inc.*"}**
- **"Microsoft Corp." = {"*Microsoft*", "*Corp.*"}**
- **"Coca Cola Co." = {"*Coca*", "*Cola*", "*Co.*"}**

Although textually dissimilar, they are structurally identical. Concretely, "*Apple*", "*Microsoft*", "*Coca*", and "*Cola*" all contain only alphabetical letters with the first one capitalized; Tokens "*Inc.*", "*Corp.*", and "*Co.*" all are alphabetical letters with first letter capitalized, and they all end with a dot. Therefore, "*Apple Inc.*" and "*Microsoft Corp.*" have the same sequence of structure vectors. Moreover, for consecutive tokens with identical structure vectors, we combine them into one and hence "*Coca Cola*" shares the same structure vectors with the other two. Therefore, if one of the three is labeled as ⟨name⟩⟨suffix⟩, we can apply the same sequence of labels to the other two examples as weak labels without actual human annotation.

To some extend, the structure vector-based weak supervision approach adopted in our framework is similar to the labeling functions/rules adopted in data programming approaches (e.g., (Ratner et al., 2017)). In our framework, predefined boolean predicates can be viewed as token-level labeling functions, which are later automatically combined as entity-level labeling functions (together with condensed BERT embeddings) used by the second MLP in our architecture (see Figure 1). Moreover, in our framework, the labeling functions are transparent to the user, thus no programming skills are needed.

2.2 Active Sampling Strategy

The model learning process has multiple iterations, where each starts with requesting the user to label the entity with highest *informative score* (to be defined shortly). Based on the user labeled entity, a set k of other entities with identical sequence of structure vectors will be automatically labeled and used for incrementally updating the model being learned. Then, unlabeled entities are annotated by the refined model and ranked according to the probability scores produced by the CRF layer. Subsequently, both top-p high-confidence and bottom-q low-confidence machine-label entities are sent to the user for verification (i,e, correct or incorrect). We also update the unlabeled entity set by removing user labeled entities and weakly labeled entities. We repeat the process until either user's labeling budget is completed or most (e.g, $\geq 90\%$) of the low-confidence labeled entities are correct.

Informative Score. The informativeness of an entity is measured according to its *representativeness* and *uncertanty*. Let $\mathcal{S}(\mathcal{E}_i)$ denote the sequence of structure vectors of entity $\mathcal{E}_i$, then we define the representativeness of $\mathcal{E}_i$ with respect to the current

Type	# entities	Components
PER	1302	⟨title⟩⟨first⟩⟨middle⟩⟨last⟩ ⟨suffix⟩⟨degree⟩
ORG	2209	⟨corename⟩⟨type⟩⟨suffix⟩⟨location⟩
DATE	1190	⟨Year⟩⟨MonthOfYear⟩⟨Day⟩
LOG	1323	⟨host⟩⟨time⟩⟨filename⟩⟨operation⟩ ⟨requesttype⟩⟨errormsg⟩⟨remainder⟩

Table 2: Statistics of datasets

set E^u of unlabeled entity as follows:

$$\text{Rep}(\mathcal{E}_i) = \mid \{\mathcal{E}_k \mid \mathcal{S}(\mathcal{E}_k) = \mathcal{S}(\mathcal{E}_i), \forall \mathcal{E}_k \in E^u\} \mid$$

Intuitively, the representativeness of an entity is the total number of entities in the unlabeled data that have the same sequence of structure vectors. The uncertainty score of an entity $\mathcal{E}_i$ is defined as:

$$\text{Uncertain}(\mathcal{E}_i) = \frac{1}{Pr(\mathcal{M}(\mathcal{E}_i))/|\mathcal{E}_i|}$$

where $Pr(\mathcal{M}(\mathcal{E}_i))$ is the probability score of the most likely sequence of labels for $\mathcal{E}_i$ produced by the final CRF layer, and $|\mathcal{E}_i|$ is the number of tokens in $\mathcal{E}$ (divided by this term to normalize the probability score wrt the length of the entities). Then, the informative score of an entity $\mathcal{E}_i$ is:

$$\text{Info}(\mathcal{E}_i) = \text{Rep}(\mathcal{E}_i) \times \text{Uncertain}(\mathcal{E}_i).$$

Thus, informative examples are the ones that are structurally highly representative and for which the current model is highly uncertain.

3 Experimental Evaluation

We implemented the system with Pytorch (Paszke et al., 2019) and pytorch-transformer (Wolf et al., 2019). Four different entity types were considered (see Table 2). Sample mentions of each entity type and corresponding expected structured representations are given in Table 3. Two baselines: (1) **CRF-AW** (Okazaki, 2007): linear-chain conditional random field using our structure vectors as features (2) **LUSTRE**: the prior SoTA active learning system for learning structured representations of entity names (Bhutani et al., 2018). More details about datasets, the implementation of the system, best-performing hyperparameter settings, and evaluation metrics can be found at https://github.com/System-T/PARTNER.

For our system, we ask the user to label the example with highest informative score (e.g., label "*Michael*" as ⟨first⟩ and "*Jordan*" as ⟨last⟩) in each active learning iteration. Then, $k = 50$ (a hyperparameter) structurally similar examples will be automatically labeled. In each iteration, 51 new labeled examples (or less, since there may not be 50 structurally similar examples) will be collected and used to incrementally refine the model. Since CRF-AW is fully supervised, we give it the sets of labels we iteratively accumulated during the active learning of our model. Hence, CRF-AW is not vanilla CRF models, they are enhanced by our structure vectors, our active learning, and our weak supervision strategies.

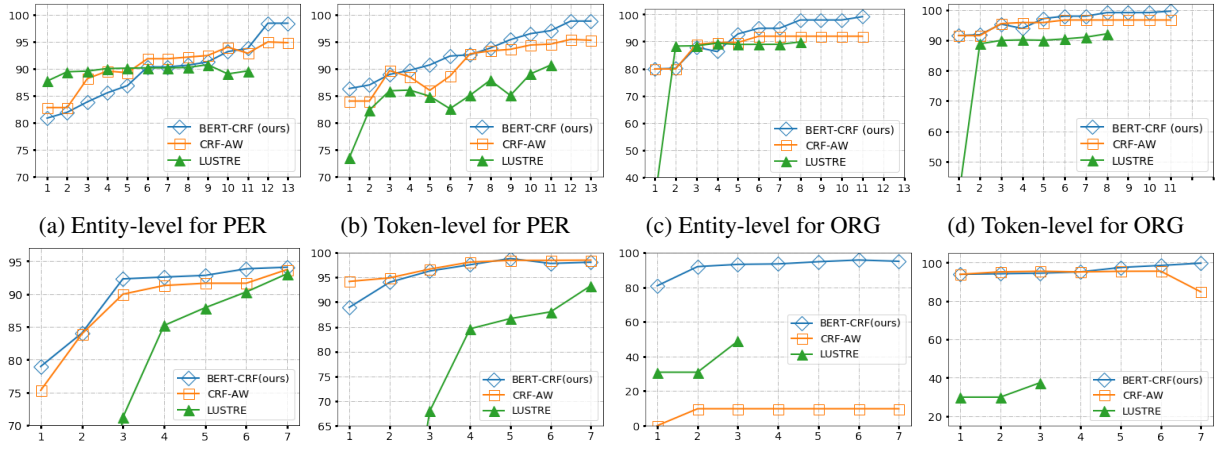
Metrics. We report F1-scores at *entity-level*, *token-level*, and *component-level*. Entity-level measures how well the model correctly labels individual entities (all tokens of an entity must be correctly labeled). For token and component-level results, we apply models to make predictions for each token in the given set of test entities. Each token prediction is credited as correct if it matches the true label. The difference between the token and component-level evaluation is that the former accumulates the credits over all tokens regardless the actual classes they belong to, whereas the latter evaluation accumulates the credits with respect to the actual classes.

Results. Figure 2 reports entity-level and token-level results for all methods at different iterations. As can be seen, our approach consistently outperforms the baselines, requiring only 7 to 13 actual user annotations per task. Moreover, as the active learning goes, the F1-score curves of our method in all tasks increase monotonically, showing stable performance. Supported by the weak labels obtained by our active sampling strategy, CRF-AW gives the suboptimal results (except for the entity-level performance for LOG), but there are still noticeable gaps between CRF-AW and ours, indicating that pre-trained BERT still plays an essential role. LUSTRE fails to match our performance except for DATE. This finding is not surprising: since LUSTRE learns highly precise rules from user labeled examples, its recall is largely determined by its sampling strategy, which is less effective to find a variety of structurally diverse examples. Since our method always make a prediction, recall is trivially 100%, but the overall precision for entity-level and token-level is relatively low initially. Then, whether or not our sampling strategy can keep finding the most informative examples to "complete" the training set is crucial for enhancing the overall precision. The monotonically increasing F1 curves indicate confirm the effectiveness of our method.

The LOG dataset consists of entities with long text and complex structures. For this dataset, CRF-AW performs well at token level but bad at entity

Type	Sample Mentions	Structured Representation
PER	*Prof Liat Sossove*	⟨title⟩⟨first⟩⟨last⟩
	Hagop Youssoufia, B.S.	⟨first⟩⟨last⟩,⟨degree⟩
	ElmeDxsna Adzemovi Sr.	⟨first⟩⟨last⟩⟨suffix⟩
ORG	*SONY CORP.*	⟨corename⟩⟨suffix⟩
	JONES APPAREL GROUP INC	⟨corename⟩⟨corename⟩⟨type⟩⟨suffix⟩
	STAPLES, INC.	⟨corename⟩,⟨suffix⟩
DATE	*February 2, 2019*	⟨MonthOfYear⟩⟨Day⟩,⟨Year⟩
	6/13/2012	⟨MonthOfYear⟩/⟨Day⟩/⟨Year⟩
	1st day of April 2019	⟨Day⟩⟨tok⟩{2}⟨MonthOfYear⟩⟨Year⟩
LOG	*719+1: Tue Aug 22 08:26:41 1995 (/wow/wow-mbos.gif): Sent binary: GET /wow/wow-mbos.gif HTTP/1.0*	⟨host⟩⟨timestamp⟩(⟨filename⟩)⟨operation⟩: ⟨requestType⟩ ⟨filename⟩ ⟨remainder⟩

Table 3: Sample entity mentions and their expected structured representations from each dataset



(a) Entity-level for PER (b) Token-level for PER (c) Entity-level for ORG (d) Token-level for ORG

(e) Entity-level for DATE (f) Token-level for DATE (g) Entity-level for LOG (h) Token-level for LOG

Figure 2: F1-scores at entity and token levels (X-axis: # iterations (i.e., # actual user labels); Y-axis: F1-scores)

		Ours	CRF-AW	LUSTRE
PER	⟨first⟩	**0.989**	0.958	0.97
	⟨middle⟩	**0.88**	0.835	0.81
	⟨last⟩	**0.975**	0.967	0.92
	⟨suffix⟩	0.761	0.70	**0.821**
	⟨nickname⟩	**0.989**	0.80	0.23
	⟨degree⟩	**0.991**	0.972	0.787
	⟨title⟩	**1**	0.765	0.652
ORG	⟨corename⟩	**0.996**	0.963	0.971
	⟨suffix⟩	**0.969**	0.792	0.923
	⟨location⟩	**0.976**	0.976	0.78
	⟨type⟩	**0.991**	0.0	0.243
DATE	⟨MonthOfYear⟩	**0.988**	0.964	0.942
	⟨Day⟩	**0.941**	0.939	0.916
	⟨Year⟩	**0.957**	**0.957**	0.944
LOG	⟨Host⟩	**1**	0.994	0.486
	⟨time⟩	**1**	**1**	0.489
	⟨filename⟩	**0.998**	0.756	0.560
	⟨operation⟩	**0.993**	0.739	0.520
	⟨remainder⟩	**0.994**	0.823	0.529
	⟨errorMessage⟩	**0.991**	0.239	0
	⟨requestType⟩	**1**	0.603	0.527

Table 4: Final F1-scores for different components

level, indicating that it can correctly label most tokens in an entity name, but makes minor mistakes leading to entity-level errors. LUSTRE outperforms CRF-AW since it learns very precise rules. However, LUSTRE terminated quickly as it runs out of memory when trying to learn a rule with over 70 regex primitives to capture a long log message.

Regarding the component-level results, as shown in Table 4, our methods significantly outperform other baselines, which is not a surprise given that our method gives the best token-level results.

4 Concluding Remarks

We proposed a framework for learning structured representation of entity names under low-resource settings. In particular, we focus on a challenging scenario, where entity names are given as textual strings without context. Experiments show the efficacy of our approach. One immediate future work is to generate explanations for model predictions using structured vector.

Acknowledgments

We would like to thank the anonymous reviewers for their critical reading and constructive suggestions, which improve and clarify this paper.

References

Arvind Arasu and Raghav Kaushik. 2009. A grammar-based entity representation framework for data cleaning. In *SIGMOD'09*, pages 233–244, New York, NY, USA. ACM.

Nikita Bhutani, Kun Qian, Yunyao Li, HV Jagadish, Mauricio Hernandez, and Mitesh Vasa. 2018. Exploiting structure in representation of named entities using active learning. In *COLING 2018*, pages 687–699.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Clemens-Alexander Brust, Christoph Käding, and Joachim Denzler. 2020. Active and incremental learning with weak supervision. *KI-Künstliche Intelligenz*, pages 1–16.

Adriano Crestani Campos, Yunyao Li, Sriram Raghavan, and Huaiyu Zhu. 2015. Entity variant generation and normalization. US Patent 9,063,926.

Zheng Chen and Heng Ji. 2011. Collaborative ranking: A case study on entity linking. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, page 771–781, USA. Association for Computational Linguistics.

Marco Dinarelli and Sophie Rosset. 2012. Tree representations in probabilistic models for extended named entities detection. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 174–184, Avignon, France. Association for Computational Linguistics.

Jenny Rose Finkel and Christopher D Manning. 2009. Nested named entity recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 141–150. Association for Computational Linguistics.

Sairam Gurajada, Lucian Popa, Kun Qian, and Prithviraj Sen. 2019. Learning-based methods with human-in-the-loop for entity resolution. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 2969–2970, New York, NY, USA. Association for Computing Machinery.

Hangfeng He and Xu Sun. 2017. A unified model for cross-domain and semi-supervised named entity recognition in chinese social media. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Meizhi Ju, Makoto Miwa, and Sophia Ananiadou. 2018. A neural layered model for nested named entity recognition. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1446–1459, New Orleans, Louisiana. Association for Computational Linguistics.

Jungo Kasai, Kun Qian, Sairam Gurajada, Yunyao Li, and Lucian Popa. 2019. Low-resource deep entity resolution with transfer and active learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5851–5861, Florence, Italy. Association for Computational Linguistics.

Arzoo Katiyar and Claire Cardie. 2018. Nested named entity recognition revisited. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 861–871, New Orleans, Louisiana. Association for Computational Linguistics.

Haodi Li, Qingcai Chen, Buzhou Tang, Xiaolong Wang, Hua Xu, Baohua Wang, and Dong Huang. 2017. Cnn-based ranking for biomedical entity normalization. *BMC bioinformatics*, 18(11):79–86.

Pierre Lison, Aliaksandr Hubin, Jeremy Barnes, and Samia Touileb. 2020. Named entity recognition without labelled data: A weak supervision approach. *arXiv preprint arXiv:2004.14723*.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2:231–244.

David Nadeau. 2007. *Semi-supervised named entity recognition: learning to recognize 100 entity types with little supervision*. Ph.D. thesis, University of Ottawa.

Naoaki Okazaki. 2007. Crfsuite: a fast implementation of conditional random fields (crfs).

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Kun Qian, Nikita Bhutani, Yunyao Li, HV Jagadish, and Mauricio Hernandez. 2018. Lustre: An interactive system for entity structured representation and variant generation. In *ICDE*, pages 1613–1616.

Kun Qian, Lucian Popa, and Prithviraj Sen. 2017. Active learning for large-scale entity resolution. In *CIKM*, pages 1379–1388. ACM.

Kun Qian, Lucian Popa, and Prithviraj Sen. 2019. Systemer: a human-in-the-loop system for explainable entity resolution. *Proceedings of the VLDB Endowment*, 12(12):1794–1797.

Kun Qian, Poornima Chozhiyath Raman, Yunyao Li, and Lucian Popa. 2020. Partner: Human-in-the-loop entity name understanding with deep learning. In *AAAI*, pages 13634–13635.

Alexander Ratner, Stephen H. Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. *Proc. VLDB Endow.*, 11(3):269–282.

Esteban Safranchik, Shiying Luo, Stephen H Bach, Elaheh Raisi, Stephen H Bach, Stephen H Bach, Daniel Rodriguez, Yintao Liu, Chong Luo, Haidong Shao, et al. 2020. Weakly supervised sequence tagging from noisy rules. In *AAAI*, pages 5570–5578.

Burr Settles. 2009. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

Yanyao Shen, Hyokun Yun, Zachary C. Lipton, Yakov Kronrod, and Animashree Anandkumar. 2018. Deep active learning for named entity recognition. In *ICLR 2018*.

Paul Thompson and Christopher C. Dozier. 1997. Name searching and information retrieval. In *Second Conference on Empirical Methods in Natural Language Processing*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Sendong Zhao, Ting Liu, Sicheng Zhao, and Fei Wang. 2019. A neural multi-task learning framework to jointly model medical named entity recognition and normalization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 817–824.

A Appendix

In the appendix, we provide more details about the experimental evaluation presented in the main paper. Other useful materials are also included in the github repository at `https://github.com/System-T/PARTNER`.

A.1 Datasets

We studied four different datasets, which will be made publicly available in the github repo mentioned earlier after we finish the open source approval process.

In our experiments, each dataset contains a training set and a held-out test set, which we will release. The datasets are manually annotated, and we adopted the 70%-30% splitting convention. Concretely, we first label all examples, and then 70% of them became the "unlabeled" data that is going to be provided to our system, and the rest 30% became a held-out test set.

Input Representation	
BERT-CRF word embeddings size	768
Input dropout rate	0
Structure Vectors	
size of predicates	15
Multilayer Perceptron Layer	
MLP-1 (after BERT) (input, output)	(768,50)
MLP-2 (after Concatenation) (input, output)	(50, # labels)
Activation function	relu
Training	
Optimization	SGD
# epochs	30
Learning rate	0.01
Learning rate decay	1×10^{-4}
Loss function	Negative log likelihood
Active Learning & Weak Supervision Parameters	
# structurally similar examples	50
# high-confidence examples	15
# low-confidence examples	15

Table 5: Architecture, hyperparameters, and training

A.2 Model Implementation

Table 5 lists the hyperparameters of the best-performing model that we reported in the main paper. As mentioned earlier, we used huggineFace pytorch-transformer (Wolf et al., 2019) to implement our BERT-CRF model. In particular, we used the pretrained `BertModel`, and we obtained the tokenizer and pretrained weights using the `bert-base-cased` configuration.

The Bert embedding size (i.e., 768) is predefined by the Bert model, and the size of predicates for structural vectors are predefined by us. The input-output dimension size of MLP-1 (bounds are determined by the embedding size of Bert and number of labels) and learning rate are determined using random sampling (using the DATE dataset).

A.3 Evaluation Metrics

We used three metrics in the main paper: *entity-level*, *token-level*, *component-level*. Here we give a concrete example for computing the three metrics. Consider a test set consisting of a single DATE entity string: **June 3rd, 2020**. Assume that we have three models: m_1, m_2, and m_3. The predictions of the three models over the single test date entity is shown in Table 6, and the F1-scores of these models at entity-level, token-level, and component-level are shown Table 7.

Given that m_1 correctly predict all the tokens (thus the entire entity), it is easy to see that it's F1-scores are all 1.0's. Similar argument can also show that m_3's F1-scores are all 0's. For m_2, since it does not correctly label all the three tokens of the

	June	3rd	2020
Ground truth	**month**	**day**	**year**
m_1's predictions	month	day	year
m_2's predictions	month	month	year
m_3's predictions	none	none	none

Table 6: Predictions of three dummy models (none means the model does not make a prediction)

	Entity-level	Token-level	Component-level	
m_1	1.0	1.0	month	1.0
			day	1.0
			year	1.0
m_2	0.0 (precision = 0) (recall = 1.0)	0.80 (precision = 0.67) (recall = 1.0)	month	0.67 (precision=0.5) (recall = 1.0)
			day	0 (precision=1.0) (recall = 0.0)
			year	1.0
m_3	0.0	0.0	month	0.0
			day	0.0
			year	0.0

Table 7: F1-scores of the three dummy models at entity-level, token-level, and component-level

entity string, so the entity-level is 0. However, it does correctly labeled two tokens (i.e., "June" and "3rd") out of the three tokens, so the precision is $0.67 = 2/3$. Moreover, since m_2 makes predictions for all the three tokens, thus the recall is trivially 100%, which means it's token-level F1 is 0.80. In fact, DL-based approaches such as ours that takes a sequence of tokens as input, will trivially achieve 100% recall. For component-level evaluation, m_2 predicted two tokens as **month**, where only one of them is correct, so the precision for month component is 50%. Since m_2 identified all the true **month** tokens, the recall is 100%. For **day** component, m_2 does not predict any token as **day**, but the second token "3rd" has true label **day**, so the recall is 0%, which leads to an F1-score of 0. We do not discuss its **year** component performance because it is obvious 100%.

A.4 Training

Unlike typical deep learning-based supervised learning approaches, where there are a lot of labeled examples, we have limited training data in each active learning iteration. It does not make sense to split the limited number of labeled examples (e.g., about 30) into a training set and a development set, and use the development set to choose the best-performing model. First, the number of examples is too small to make the splitting meaningful. Second, we could potentially perform k-fold cross validation, but that would require much more time, which makes the user experience bad (i.e., the user has to wait for a long time before the training is done).

To make the system as interactive as possible, we used a simple heuristic, that is, we simply train the model with a fixed number of epoch (with random shuffling after each epoch), in our experiments, we set the number to 30 epochs. However, we would terminate the training early if the difference between the total loss of two consecutive epochs is less than a certain threshold, which is set to 10^{-3} in our experiments. The main intuition is that we want to let the model somewhat overfit the training data as they are considered to be "informative" based on our active learning strategy, but we do need to avoid "extreme" overfitting.

A.5 Environment and Runtime

We have run our experiments both on a CPU machine (Apple Macbook Pro 2019 model) and on a GPU machine (with 1 Tesla V100 GPU). Recall that our framework is active learning-based, where each active learning iteration contains three steps: (1) user labeling, (2) model updating, and (3) user feedback. The most time-consuming part is the model updating phase. For the experiments reported in this paper, depending on the sizes of labeled data, the model updating phase takes 10 to 70 seconds with the GPU machine, and 30 seconds to 10 mins with the CPU machine.