On the Interactions of Structural Constraints and Data Resources for Structured Prediction

Zhisong Zhang, Emma Strubell, Eduard Hovy
Language Technologies Institute, Carnegie Mellon University
zhisongz@cs.cmu.edu, strubell@cmu.edu, hovy@cmu.edu

Abstract

In this work, we provide an analysis on the interactions of the effectiveness of decoding with structural constraints and the amount of available training data for structured prediction tasks in NLP. Our exploration adopts a simple protocol that enforces constraints upon constraint-agnostic local models at testing time. With evaluations on three typical structured prediction tasks (named entity recognition, dependency parsing, and event argument extraction), we find that models trained with less data predict outputs with more structural violations in greedy decoding mode. Incorporating constraints provides consistent performance improvements and such benefits are larger in lower resource scenarios. Moreover, there are similar patterns with regard to the model sizes and more efficient models tend to enjoy more benefits. Finally, we also investigate settings with genre transfer and discover patterns that are related to domain discrepancies.

1 Introduction

Recently, neural models, especially those based on pre-trained contextualized representations, have brought impressive improvements for a variety of structured prediction tasks in NLP (Devlin et al., 2019; Kulmizev et al., 2019; Shi and Lin, 2019; Li et al., 2020a). More interestingly, the incorporation of powerful neural models seems to decrease the potential benefits brought by more complex structured output modeling. For example, for sequence labeling, it has been shown that reasonably good performance could be obtained even without any explicit modeling of the interactions of the output tags (Tan et al., 2018; Devlin et al., 2019). For dependency parsing, models that ignore tree constraints and cast the problem as head selection in training can still obtain impressive results (Dozat and Manning, 2017). Most of these previous results are obtained in fully supervised settings. While they show that with abundant training signals, better input modeling and representation learning could shadow the benefits brought by more complex structured modeling, it remains unclear for the cases where data resources are limited.

One of the most salient and important properties of structured prediction is that the output objects should follow specific structural constraints. For example, the output of a syntactic parser should be a well-formed tree and the output labels of an information extraction system need to follow certain type restrictions. In this work, we focus on the facet of structural constraints and explore its influence on structured prediction problems under scenarios with different amounts of training data. On the one hand, since we know the target outputs should conform to certain constraints, *explicitly* enforcing these constraints will likely bring benefits and sometimes even be a requirement. On the other hand, as neural models are developed to better represent input contexts, they might already be able to *implicitly* capture the output constraints by learning from the data. In particular, it would be unsurprising that the model could directly produce outputs that conform to constraints without explicit enforcement, given enough training data, since the training instances are presented as such.

Regarding the interactions between explicit incorporation of constraints and the amount of training data, we ask the following three research questions (RQs), which we aim to explore in this work:

RQ1: What is the influence of constraints with different amounts of training data?
With powerful neural networks and abundant training data, the model can be trained to implicitly capture structural constraints even without explicit enforcement. Nevertheless, it still remains unclear for the cases with limited data. We aim to explore how the incorporation of constraints influences the outputs and how such influences change with dif-

Proceedings of The Fourth Workshop on Simple and Efficient Natural Language Processing (SustaiNLP), pages 147–157

ferent amounts of training data.

RQ2: What is the influence of constraints when using more efficient models?
Although neural models can obtain impressive results, one shortcoming is that they are usually computationally expensive. Recently, there have been many works on improving model efficiency. Knowledge distillation is one of the most widely-utilized methods, learning a smaller student model from a larger teacher model (Kim and Rush, 2016; Sanh et al., 2019; Jiao et al., 2020). An interesting question to explore is how these more efficient models interact with the explicit incorporation of structural constraints.

RQ3: What is the influence of constraints for out-of-domain generalization?
We usually expect the model to be able to generalize to scenarios that can be different from those represented by the training data, for example, to different domains or text genres. It will be interesting to explore how the constraints influence predictions for these cases and especially whether there are specific patterns with regard to the discrepancies between the source and the target.

To answer these questions, we conduct extensive experiments on three typical structured prediction tasks, including named entity recognition (NER), dependency parsing (DPAR) and an information extraction task of event argument extraction (EAE). We find that models trained with less training data tend to produce outputs that contain more structural violations when using constraint-agnostic greedy decoding. Further applying constrained decoding brings consistent performance improvements and the benefits are more prominent in lower data scenarios (§3.2). A similar trend can be found with regard to model size: Smaller models tend to output more violations with greedy decoding and benefit more from constrained decoding (§3.3). Finally, in cross-genre settings, we find a weak pattern with regard to genre discrepancies: More structural violations tend to be made with greedy decoding when transferring to more distant genres (§3.4).

2 Tasks and Models

2.1 Named Entity Recognition

Our first task is named entity recognition (NER), which aims to extract entity mentions from raw texts and can be typically cast as a sequence labeling problem. We adopt a simple NER model

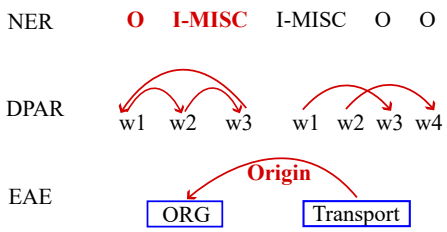

Figure 1: Examples of structural violations (marked in red). For NER, the tag transition from 'O' to 'I-MISC' is illegal. For DPAR, the left subtree contains a loop while the right one has crossing edges. For EAE, the ORIGIN role cannot be assigned to an ORG entity.

that utilizes a pre-trained BERT model as the encoder and a softmax layer to predict the output tags. We adopt the typical BIO tagging scheme (Ramshaw and Marcus, 1995), specifying tags for the **B**eginning, the **I**nside and the **O**utside of an entity span.

More specifically, for an input sequence of words $[w_1, w_2, \cdots, w_n]$, our model aims to assign a sequence of BIO tags $[t_1, t_2, \cdots, t_n]$ for them. The probability of each output tag is locally normalized for each word:

$$p(t_i|w_i) = \frac{\exp \text{score}(t_i|w_i)}{\sum_{t' \in \mathcal{T}} \exp \text{score}(t'|w_i)}$$

Here, the score$(\cdot)$ function is realized as a linear layer stacked upon the word representations[1] and $\mathcal{T}$ denotes the output tag space.

With the BIO tagging scheme, there are hard constraints between tags of consecutive tokens: The **I** tag must follow a **B** or **I** tag of the same entity type. For example, the tagged sequence "O I-MISC I-MISC O O" is erroneous because the transition "O $\rightarrow$ I-MISC" is illegal. One solution to mitigate this problem is to forbid such illegal transitions in decoding. This can be achieved by incorporating a transition matrix $M \in \mathcal{R}^{|\mathcal{T}| \times |\mathcal{T}|}$, where the entries corresponding to illegal tag transitions are filled with $-\infty$ and the legal ones are filled with 0. For the decoding process, we define the score of a tag sequence as:

$$s(t_1, t_2, \cdots, t_n) = \sum_i \log p(t_i|w_i) + \sum_i M_{t_i, t_{i+1}}$$

In this way, the highest scoring tag sequence will not contain transition violations. This decoding

[1] If a word is split into multiple tokens, we simply take its first sub-token.

problem can be solved efficiently by the Viterbi algorithm (Viterbi, 1967). If not enforcing these constraints, the second term of the sequence score can be dropped and the decoding will be greedily finding the maximally-scored tag for each token individually.

Notice that this treatment resembles conditional random field (CRF) based models (Lafferty et al., 2001), wherein the main difference is that we utilize a locally normalized model and the transition matrix is manually specified to exclude illegal transitions. In our preliminary experiments, we also tried CRF models but did not find obvious benefits compared to local models when adopting the same underlying pre-trained model.

2.2 Dependency Parsing

We further consider dependency parsing (DPAR) (Kübler et al., 2009), which aims to parse the input sentence into well-formed tree structures. We adopt the widely utilized first-order graph-based parser (McDonald et al., 2005). Similar to NER, we adopt the pre-trained BERT encoder to provide the contextualized representations for the input tokens and stack a biaffine scorer (Dozat and Manning, 2017) to assign scores for the dependency edges. For training, we adopt a local model that views the problem as a head-finding classification task for each input token (Dozat and Manning, 2017; Zhang et al., 2017). At testing time, we further consider tree constraints with specific decoding algorithms. Since we are mainly interested in structural tree constraints, we only perform unlabeled parsing.

More specifically, for an input sequence of words $[w_1, w_2, \cdots, w_n]$, we aim to find the dependency head words $[h_1, h_2, \cdots, h_n]$ for the input word sequence. With local normalization, this can be viewed as a head classification problem:

$$p(h_i|w_i) = \frac{\exp \text{score}(h_i|w_i)}{\sum_{h' \in \{R, w_1, w_2, \cdots, w_n\}} \exp \text{score}(h'|w_i)}$$

Here we add an artificial target R to the output space to cover the case of root nodes. The score$(\cdot)$ function is realized with a biaffine module that produces head-modifier scores for the input pair of words.

We consider two constraints for the output structures. First, there should not be any cycles in the output graphs, otherwise, they will not be trees. Moreover, we consider the projectivity constraint,[2] which specifies that there are no edges that cross each other. We adopt Eisner's algorithm (Eisner, 1996) for the constrained decoding, which is a dynamic programming algorithm that searches the highest scored trees in the constrained output space. If not considering any of these constraints, we greedily predict the head word for each token based on the head classification probabilities.

2.3 Event Argument Extraction

Finally, we consider event argument extraction (EAE), an information extraction task that aims to extract arguments for the event mentions from the texts (Ahn, 2006). For a pair of event trigger and entity mention, this task aims to link them with an argument role indicating that the entity can play such a role in the event frame. If no such role is possible, then no links are added. We again adopt a pre-trained BERT encoder for encoding and further stack a task-specific predictor, which is a biaffine scorer, similar to dependency parsing. The main difference is that here we perform local normalization for each event-entity pair since there are no constraints on how many other mentions that one mention can be linked to for event argument extraction. To better explore real application scenarios, we train an extra sequence labeler to extract event and entity mentions rather than using gold mentions. This mention detection model is the same as the one described in our NER experiments.

More specifically, our model takes a pair of event trigger and entity mention (m_t and m_e) and assigns the probabilities of argument roles to them:

$$p(r|m_t, m_e) = \frac{\exp \text{score}(r|m_t, m_e)}{\sum_{r' \in \mathcal{R} \cup \{\epsilon\}} \exp \text{score}(r'|m_t, m_e)}$$

Here, $\mathcal{R}$ denotes the role labeling space and we further include an option of ϵ to denote there are no argument relations between the event trigger and entity mention. The score function is realized with a biaffine module that produces argument scores for the input mention pair. Since a mention may contain multiple words, we concatenate the word representations of the starting and ending words to form the mention's input vector.

In event extraction, there are constraints on the mention (event and entity) types and argument role

[2] We only perform experiments on English, which is a highly projective language. Extensions to non-projective languages are left to future work.

Data	Split	#Sent.	#Token	#Event	#Entity	#Argument	#Relation
CoNLL03	train	14.0K	203.6K	-	23.5K	-	-
	dev	3.3K	51.4K	-	5.9K	-	-
	test	3.5K	46.4K	-	5.6K	-	-
UD-EWT	train	12.5K	204.6K	-	-	-	-
	dev	2.0K	25.1K	-	-	-	-
	test	2.1K	25.1K	-	-	-	-
ACE05	train	14.4K	215.2K	3.7K	38.0K	5.7K	6.2K
	dev	2.5K	34.5K	0.5K	6.0K	0.7K	0.8K
	test	4.0K	61.5K	1.1K	10.8K	1.7K	1.7K

Table 1: Data statistics of the datasets utilized in our main experiments.

labels. For example, the PERSON role of a MARRY event should have the entity type of PER, while the DESTINATION or ORIGIN roles of a TRANSPORT should have entity types denoting places (GPE, LOC or FAC). We adopt a simple method to incorporate such constraints in decoding by ignoring (masking out) the roles that are not possible according to the event and entity types. The role constraints are manually collected according to the event annotation guideline (LDC, 2005). If not considering these role constraints, we simply adopt greedy prediction for each event-entity pair.

3 Experiments

3.1 Settings

Data. Our experiments are conducted on widely utilized English datasets. In our main experiments, we adopt the CoNLL-2003 English dataset[3] (Tjong Kim Sang and De Meulder, 2003) for NER and the English Web Treebank (EWT) from Universal Dependencies[4] v2.10 (Nivre et al., 2020) for DPAR. In the genre transfer experiments for NER and DPAR, we utilize OntoNotes 5.0 dataset[5] (Weischedel et al., 2013) and split the data according to text genres. For the event task, we adopt the ACE05 dataset[6] (Walker et al., 2006), using the scripts from Lin et al. (2020) for the pre-processing.[7] Table 1 shows the data statistics.

Model and training. Unless otherwise specified, we adopt the pre-trained $\text{BERT}_{\text{base}}$ as the contextualized encoder for our models. The encoder is fined-tuned with the task-specific decoders in all the experiments. The number of model parameters is around 110M. We follow common practices for the settings of other hyper-parameters. Adam (Kingma and Ba, 2014) is utilized as the optimizer. The learning rate is initially set to 1e-5 for NER and 2e-5 for DPAR and EAE. It is further linearly decayed to 10% of the initial value throughout the training process. The models are trained for 20K steps with a batch size of around 512 tokens. We pick final models by the performance on the development set of each task. The original development sets are also down-sampled accordingly as the training sets to simulate scenarios with different data amounts. All the reported results are averaged over five runs with different random seeds.

Task	Model	5K	20K	100K
NER	Local	$84.27_{0.8}$	$88.91_{0.6}$	$91.24_{0.3}$
	Global	$84.73_{0.1}$	$88.92_{0.4}$	$91.38_{0.2}$
DPAR	Local	$84.57_{0.1}$	$89.46_{0.2}$	$91.95_{0.1}$
	Global	$82.65_{0.3}$	$88.92_{0.3}$	$91.65_{0.2}$

Table 2: Comparisons between local and global models for NER (F1%) and DPAR (UAS%). Numbers in the subscripts denote standard deviation.

Local normalization. In our main experiments, we choose locally normalized models instead of more complex global models. Table 2 provides comparisons between the local and global models for NER and DPAR. For the global models, we use a standard linear-chain CRF (Lafferty et al., 2001) for NER and tree-CRF (Paskin, 2001) for DPAR. For these results, constrained decoding is applied since it is found to be helpful for both local and global models. The results show that there are no clear benefits of using global models over the simpler local models, probably due to the strong input context modeling capabilities of the underlying pre-trained encoders. Therefore, we simply adopt local models in our main experiments.

[3] https://www.clips.uantwerpen.be/conll2003/ner/
[4] https://universaldependencies.org/
[5] https://catalog.ldc.upenn.edu/LDC2013T19
[6] https://catalog.ldc.upenn.edu/LDC2006T06
[7] http://blender.cs.illinois.edu/software/oneie/

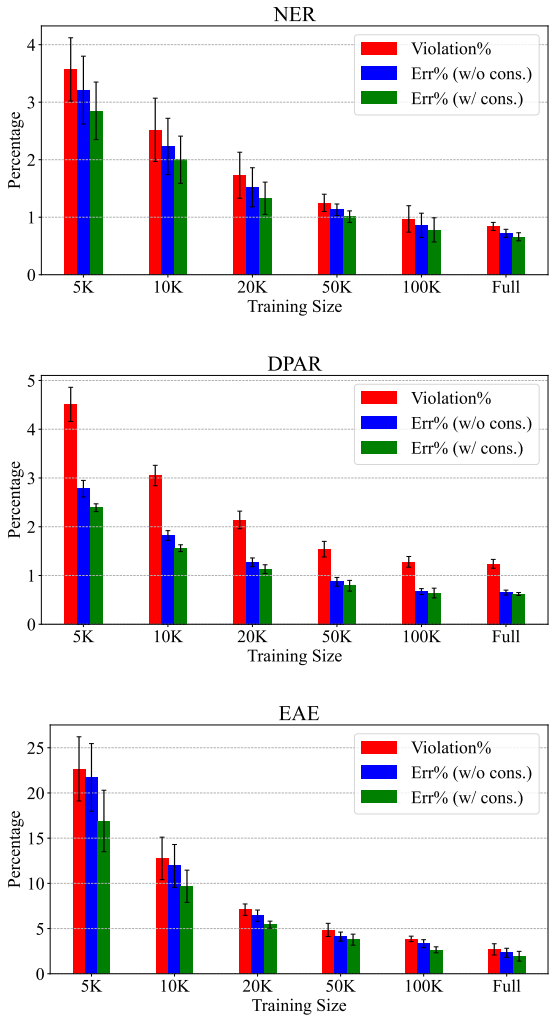

Figure 2: Illustrations of constraint violations and related error rates. Here, "Violation%" denotes the percentages of predicted items that violates structural constraints in the greedy decoding mode, and "Err%" denotes the percentages of the predicted items that violate the constraints and are incorrect at the same time.

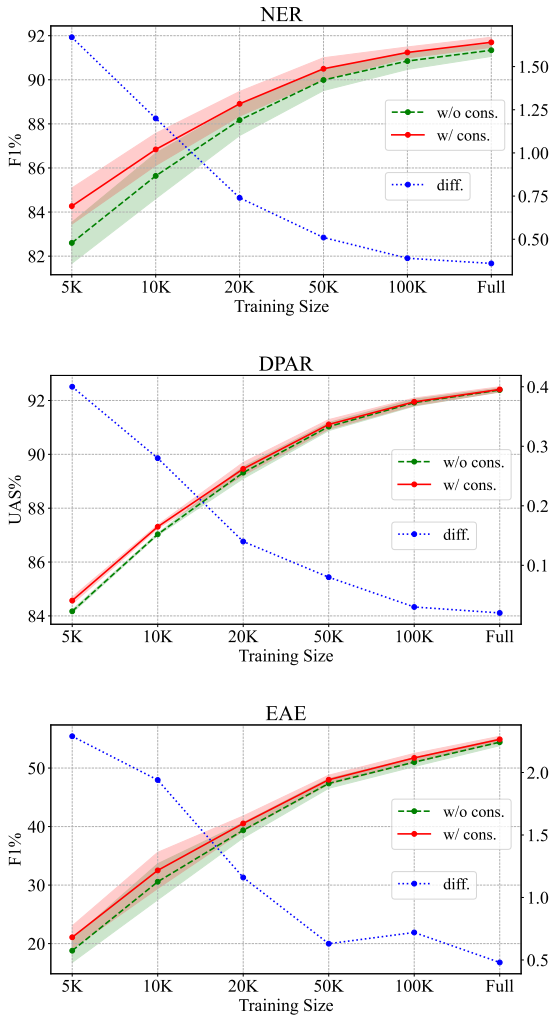

Figure 3: Test results with or without applying constraints against different training sizes. Here, x-axis denotes the training size (measured by the number of tokens). The left y-axis denotes the performance (F1% for NER, UAS% for DPAR and F1% for EAE). The right y-axis denotes the performance differences between the methods with or without constraints.

Evaluation. We adopt standard evaluation metrics for the tasks: Labeled F1 score for NER, unlabeled attachment score (UAS) for DPAR, labeled argument F1 score for EAE (Lin et al., 2020).

3.2 RQ1: On Training Data

We first investigate the effectiveness of incorporating constraints in decoding, plotting the rates of structural violations and related errors in Figure 2. For all the predicted items (all non-'O' tags for NER, all dependency edges for DPAR and all predicted argument links for EAE), we calculate the percentage of items that violate the structural constraints when using greedy decoding ("Violation%"). For NER, we analyze at the tag level and count the illegal tag transitions. For DPAR, we include the edges that are inside a loop (violating the acyclic constraint) or go across another edge (violating the projective constraint). For EAE, we count the argument links whose role does not comply with the types of the event and the entity that it connects. We further calculate "Err%", which denotes the percentage of the items that contain violations in greedy decoding and are wrongly predicted at the same time. Such error rates are calculated for both greedy (w/o cons.) and constrained (w/ cons.) modes, and the comparisons between these two can illustrate the amount of error reduction that constrained decoding can bring.

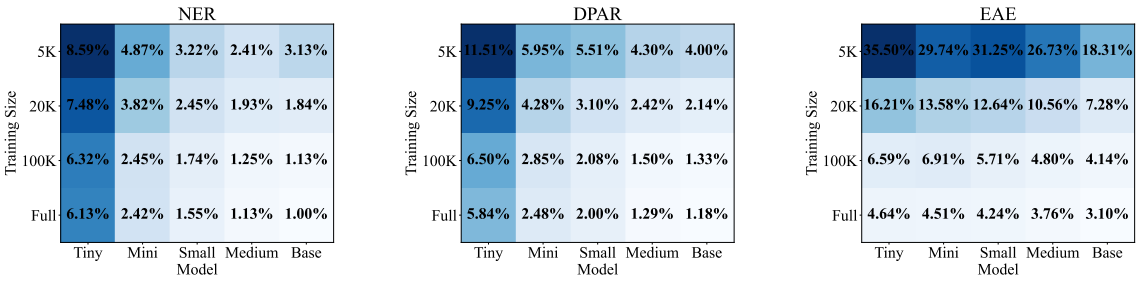

Figure 4: "Violation%" (percentages of predicted items that violates constraints with greedy decoding) with different models and amounts of training data. Here, x-axis denotes the underlying model while y-axis denotes training sizes.

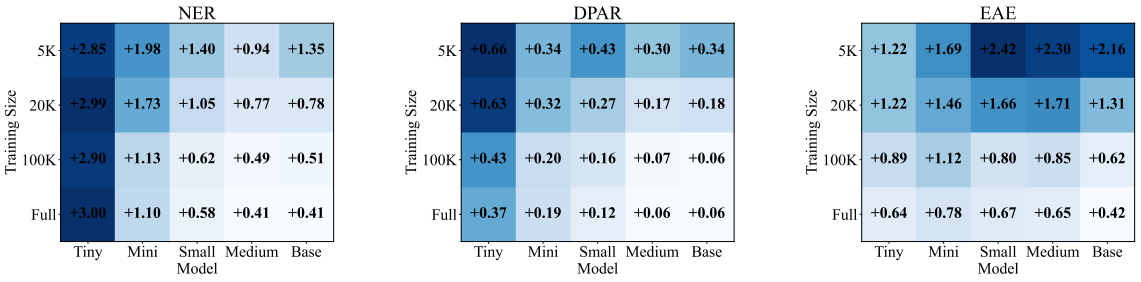

Figure 5: Performance improvements brought by constrained decoding with different models and amounts of training data. Here, x-axis denotes the underlying model while y-axis denotes training sizes.

The overall trends are consistent on all the tasks. As we have more training data, there are fewer structural violations without explicitly enforcing constraints, which indicates that the model can implicitly learn the constraints if given enough training data. Moreover, although constrained decoding can eliminate such violations, they do not always lead to the correct predictions; only a small portion of incorrect items can be corrected with constrained decoding, and such improvements are more prominent with less training data.

We further show the main test results in Figure 3. The general trends are again similar for all three tasks: Constraints provide consistent benefits for the model performance, and such benefits are larger as we have less training data. This corresponds well to the violation analysis in Figure 2: with enough training data, the model implicitly learns the structural constraints from the data and further enhancement of constrained decoding will make little difference; however, with less training data, explicitly enforcing constraints can help.

RQ1 Takeaways: Without incorporating constraints, there are more constraint violations from the predictions of the models trained with less data. By enforcing constraints in decoding, there can be consistent benefits for model performance and such improvements are greater with models learned with less training data.

3.3 RQ2: On Efficient Models

We further explore the influence of using more efficient models. We take the distilled versions of the BERT models from Turc et al. (2019) and repeat our previous experiments. Specifically, we consider five models (L=Layer Number, H=Dimension Size): Tiny (L=2, H=128), Mini (L=4, H=256), Small (L=4, H=512), Medium (L=8, H=512), and Base (L=12, H=768). We plot "Violation%" and performance differences in Figure 4 and Figure 5, respectively.

First, if looking at the axis of the training data size, the overall trends are similar to previous findings: There are more violations with less training data, and enforcing constraints helps more in lower-resource scenarios. This trend is generally consistent across all the underlying models. Moreover, comparing across the model axis brings more interesting findings. Overall, the smaller models tend to output predictions with more violations if adopting greedy decoding and incorporating constraints generally bring more performance improvements for smaller models. The reason for this trend might be that smaller models contain fewer parameters

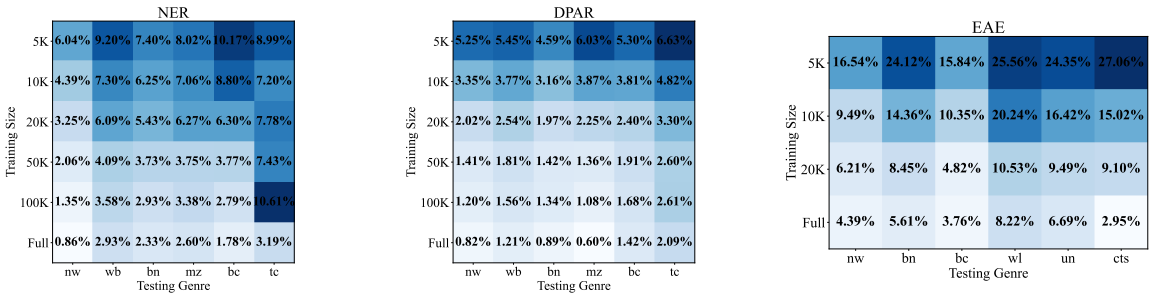

Figure 6: "Violation%" (percentages of predicted items that violates structural constraints) on different testing genres with different amounts of source training data ("nw" as the training source). Here, x-axis denotes the testing genres (which are sorted with the similarities to the source genre) while y-axis denotes training sizes.

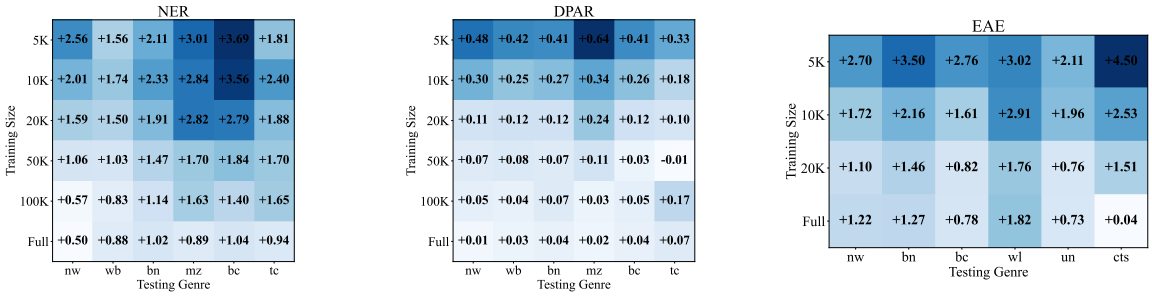

Figure 7: Performance improvements brought by constrained decoding on different testing genres with different amounts of source training data ("nw" as the training source). Here, x-axis denotes the testing genres (which are sorted with the similarities to the source genre) while y-axis denotes training sizes.

	Tiny	Mini	Small	Medium	Base
NER$_{w/o}$	0.29	0.32	0.36	0.53	1.19
NER$_{w/}$	0.56	0.59	0.64	0.80	1.45
DPAR$_{w/o}$	0.23	0.26	0.33	0.47	1.07
DPAR$_{w/}$	0.28	0.31	0.36	0.50	1.10

Table 3: Decoding speed (ms per sentence) without ($_{w/o}$) or with ($_{w/}$) constraints.

to learn all the patterns in the training data and such under-parameterization may bring difficulties in implicitly capturing the constraints.

Another interesting question is how decoding speed is influenced by the underlying model and the decoding algorithm. Table 3 presents the time required to decode one sentence for NER and DPAR. Here, we do not analyze the EAE task, since there are no complex algorithms involved for our constrained decoding for EAE and we did not find obvious speed differences between decoding methods with or without constraints. Generally, constrained decoding requires more computational cost compared with the constraint-agnostic greedy methods. This is not surprising since the constraint-agnostic decoding method simply predicts the locally maximally scored items while constrained decoding needs to invoke algorithms with higher complexity. With smaller models, constrained decoding brings relatively more cost because there are less intense computational requirements for the underlying encoder. This trend is especially obvious for the NER task, where constrained decoding costs nearly twice the time as greedy decoding when using the Tiny model. When adopting larger models, the encoder starts to require more computations and thus the relative extra cost brought by constrained decoding takes a smaller proportion.

RQ2 Takeaways: Smaller and more efficient models such as distilled versions of BERT tend to output predictions with more structural violations with greedy decoding, and constrained decoding generally brings more benefits.

3.4 RQ3: On Genre Transfer

Finally, we explore a transfer-learning scenario where there are discrepancies between the training and testing data distributions. Specifically, we consider transferring across different text genres. For these experiments, we utilize OntoNotes for NER and DPAR, and ACE05 for EAE. We take the

newswire (nw) portion as the source for training and directly test the source-trained model on the test sets of other genres (in a zero-shot manner).

The results are shown in Figure 6 and 7, where the notations are similar to those in §3.3. In these results, similar patterns along the data size axis can be found: Incorporating constraints is more helpful in the cases with less training data and such trends generally hold for out-of-distribution testing scenarios (target genres that are not "nw") as well.

Another interesting dimension is the pattern along the axis of genres. In the figures, we sort the testing genres according to their similarities to the source (nw). To calculate the similarities between genres, we use the overlapping rate of vocabularies since lexical overlaps can be one important factor for the effectiveness of transfer. Overall, there is a weak trend that when transferring to more distant genres, greedy decoding tends to produce outputs with more structural violations. However, such a pattern is not consistent across all cases, and one potential reason might be the instability of model transfer. Moreover, there can be more appropriate measurements than our simple lexicon-based similarity that may better reflect how the predictions are influenced by constrained decoding across genres. We leave more explorations to future work.

RQ3 Takeaways: The previous patterns still generally hold for testing on out-of-domain instances with genre discrepancies: Models trained with less data tend to make more violations with greedy decoding and benefit more from constrained decoding. There is also a weak pattern when transferring to more distant genres, wherein greedy decoding tends to produce more violations.

4 Related Work

For structured prediction tasks, one important property is that the prediction outputs are complex objects with multiple interdependent variables. How to model such inter-dependencies is an important question for traditional NLP research. Classical algorithms for decoding and learning have been developed for various structured prediction tasks, including the Viterbi algorithm (Viterbi, 1967) and forward-backward algorithm (Baum et al., 1970) for sequence labeling, maximum spanning tree algorithm (Chu and Liu, 1965; Edmonds, 1967), Inside-Outside algorithm (Paskin, 2001) and Matrix-Tree Theorem (Koo et al., 2007; Smith and Smith, 2007; McDonald and Satta, 2007) for dependency parsing, as well as more complex algorithms for tasks involving more complicated graph structures (Rush and Collins, 2012; Burkett and Klein, 2013; Martins et al., 2015; Gormley and Eisner, 2015). Though recent developments in neural models and pre-trained language models have boosted the performance of simple local models, better modeling of the structured outputs have still been shown effective for various structured prediction tasks (Wang et al., 2019; Fonseca and Martins, 2020; Zhang et al., 2020; Wei et al., 2021).

For the output modeling of structured prediction tasks, the hard structural constraint is a key factor for the development of decoding and learning algorithms. To enhance general explicitly stated constraints, Roth and Yih (2004) tackle the decoding problem with Integer Linear Programming (ILP) and such paradigm has been applied to a range of structured NLP tasks (Denis and Baldridge, 2007; Roth and Yih, 2007; Clarke and Lapata, 2008; Punyakanok et al., 2008). In addition to enforcing well-formed output structures for decoding, constraints can be also incorporated to enhance model learning (Chang et al., 2008; Li et al., 2020b; Pan et al., 2020; Wang et al., 2020, 2021). While we mainly focus on simply applying constrained decoding with local models trained with different amounts of data, it would be interesting to explore the influences when further incorporating constraints at model training time.

5 Conclusion

In this work, we explore the interactions of constraint-based decoding algorithms and the amounts of training data for typical structured prediction tasks in NLP. Specifically, we train local models with different amounts of training data and analyze the influence of whether to adopt constrained decoding or not. The results show that when the model is trained with less data, the predictions contain more structural violations with greedy decoding and there are more benefits on model performance by further applying constrained decoding. Such patterns also generally hold with more efficient models and when transferring across text genres, where there are further interesting patterns with regard to model sizes and genre distances.

Limitations

This work has several limitations. First, we only experiment on English datasets. It would be interesting to explore whether the general patterns hold for non-English languages with different structural properties. Moreover, we only explore incorporating hard constraints for decoding with local models at testing time. Exploring more applications of structural constraints, such as learning with constraints, or incorporating other types of constraints, such as soft ones, would be promising future directions. Finally, we only explore three simple sentence-level structured prediction tasks, while extentions can be made to more complex tasks with larger output space, such as text generation or document-level information extraction, where constraints may play more interesting roles.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8, Sydney, Australia. Association for Computational Linguistics.

Leonard E Baum, Ted Petrie, George Soules, and Norman Weiss. 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The annals of mathematical statistics*, 41(1):164–171.

David Burkett and Dan Klein. 2013. Variational inference for structured NLP models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Tutorials)*, pages 9–10, Sofia, Bulgaria. Association for Computational Linguistics.

Ming-Wei Chang, Lev-Arie Ratinov, Nicholas Rizzolo, and Dan Roth. 2008. Learning and inference with constraints. In *AAAI*, pages 1513–1518.

Y.J. Chu and T.H. Liu. 1965. On the shortest arborescence of a directed graph. *Scientia Sinica*, 14:1396–1400.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:399–429.

Pascal Denis and Jason Baldridge. 2007. Joint determination of anaphoricity and coreference resolution using integer programming. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 236–243, Rochester, New York. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *ICLR*.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards, B*, 71:233–240.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.

Erick Fonseca and André F. T. Martins. 2020. Revisiting higher-order dependency parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8795–8800, Online. Association for Computational Linguistics.

Matthew R. Gormley and Jason Eisner. 2015. Structured belief propagation for NLP. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing: Tutorial Abstracts*, pages 5–6, Beijing, China. Association for Computational Linguistics.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

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

Terry Koo, Amir Globerson, Xavier Carreras, and Michael Collins. 2007. Structured prediction models via the matrix-tree theorem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 141–150, Prague, Czech Republic. Association for Computational Linguistics.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. Dependency parsing. *Synthesis lectures on human language technologies*, 1(1):1–127.

Artur Kulmizev, Miryam de Lhoneux, Johannes Gontrum, Elena Fano, and Joakim Nivre. 2019. Deep contextualized word embeddings in transition-based and graph-based dependency parsing - a tale of two parsers revisited. 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 2755–2768, Hong Kong, China. Association for Computational Linguistics.

John D Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289.

LDC. 2005. ACE (automatic content extraction) english annotation guidelines for events version 5.4.3. *Linguistic Data Consortium*.

Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. 2020a. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):50–70.

Tao Li, Parth Anand Jawale, Martha Palmer, and Vivek Srikumar. 2020b. Structured tuning for semantic role labeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8402–8412, Online. Association for Computational Linguistics.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7999–8009, Online. Association for Computational Linguistics.

André FT Martins, Mário AT Figueiredo, Pedro MQ Aguiar, Noah A Smith, and Eric P Xing. 2015. Ad3: Alternating directions dual decomposition for map inference in graphical models. *The Journal of Machine Learning Research*, 16(1):495–545.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 91–98, Ann Arbor, Michigan. Association for Computational Linguistics.

Ryan McDonald and Giorgio Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proceedings of the Tenth International Conference on Parsing Technologies*, pages 121–132, Prague, Czech Republic. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France. European Language Resources Association.

Xingyuan Pan, Maitrey Mehta, and Vivek Srikumar. 2020. Learning constraints for structured prediction using rectifier networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4843–4858, Online. Association for Computational Linguistics.

Mark A Paskin. 2001. Cubic-time parsing and learning algorithms for grammatical bigram models.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *Third Workshop on Very Large Corpora*.

Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, pages 1–8, Boston, Massachusetts, USA. Association for Computational Linguistics.

Dan Roth and Wen-tau Yih. 2007. Global inference for entity and relation identification via a linear programming formulation. *Introduction to statistical relational learning*, pages 553–580.

Alexander M Rush and MJ Collins. 2012. A tutorial on dual decomposition and lagrangian relaxation for inference in natural language processing. *Journal of Artificial Intelligence Research*, 45:305–362.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Peng Shi and Jimmy Lin. 2019. Simple bert models for relation extraction and semantic role labeling. *arXiv preprint arXiv:1904.05255*.

David A. Smith and Noah A. Smith. 2007. Probabilistic models of nonprojective dependency trees. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 132–140, Prague, Czech Republic. Association for Computational Linguistics.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

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.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962*.

Andrew Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. ACE 2005 multilingual training corpus. *Linguistic Data Consortium*, 57.

Haoyu Wang, Muhao Chen, Hongming Zhang, and Dan Roth. 2020. Joint constrained learning for event-event relation extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 696–706, Online. Association for Computational Linguistics.

Haoyu Wang, Hongming Zhang, Muhao Chen, and Dan Roth. 2021. Learning constraints and descriptive segmentation for subevent detection. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5216–5226, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xinyu Wang, Jingxian Huang, and Kewei Tu. 2019. Second-order semantic dependency parsing with end-to-end neural networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4618, Florence, Italy. Association for Computational Linguistics.

Tianwen Wei, Jianwei Qi, Shenghuan He, and Songtao Sun. 2021. Masked conditional random fields for sequence labeling. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2024–2035, Online. Association for Computational Linguistics.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0. *Linguistic Data Consortium, Philadelphia, PA*, 23.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017. Dependency parsing as head selection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 665–676, Valencia, Spain. Association for Computational Linguistics.

Yu Zhang, Zhenghua Li, and Min Zhang. 2020. Efficient second-order TreeCRF for neural dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3295–3305, Online. Association for Computational Linguistics.