Constrained Decoding for Computationally Efficient Named Entity Recognition Taggers

Brian Lester, Daniel Pressel, Amy Hemmeter,
Sagnik Ray Choudhury, and Srinivas Bangalore
Interactions, Ann Arbor MI 48104
{blester,dpressel,ahemmeter,schoudhury,sbangalore}
@interactions.com

Abstract

Current state-of-the-art models for named entity recognition (NER) are neural models with a conditional random field (CRF) as the final layer. Entities are represented as per-token labels with a special structure in order to decode them into spans. Current work eschews prior knowledge of how the span encoding scheme works and relies on the CRF learning which transitions are illegal and which are not to facilitate global coherence. We find that by constraining the output to suppress illegal transitions we can train a tagger with a cross-entropy loss twice as fast as a CRF with differences in F1 that are statistically insignificant, effectively eliminating the need for a CRF. We analyze the dynamics of tag co-occurrence to explain when these constraints are most effective and provide open source implementations of our tagger in both PyTorch and TensorFlow.

1 Introduction

Named entity recognition (NER) is the task of finding phrases of interest in text that map to real world entities such as organizations ("ORG") or locations ("LOC"). This is normally cast as a sequence labeling problem where each token is assigned a label that represents its entity type. Multi-token entities are handled by having special "Beginning" and "Inside" indicators that specify which tokens start, continue, or change the type of an entity. Ratinov and Roth (2009) show that the IOBES tagging scheme, where entity spans must begin with a "B" token, end with an "E" token and where single token entities are labeled with an "S", performs better than the traditional BIO scheme. The IOBES tagging scheme dictates that some token sequences are illegal. For example, one cannot start an entity with an "E" tag (such as a transition from an "O", meaning it is outside of an entity, to "E-ORG") nor can they change types in the middle of an entity—for example, transitioning from "I-ORG" to "I-LOC". Most approaches to NER rely on the model learning which transitions are legal from the training data rather than injecting prior knowledge of how the encoding scheme works.

It is conventional wisdom that, for NER, models with a linear-chain conditional random field (CRF) (Lafferty et al., 2001) layer perform better than those without, yielding relative performance increases between 2 and 3 percent in F1 (Ma and Hovy, 2016; Lample et al., 2016). A CRF with Viterbi decoding promotes, but does not guarantee, global coherence while simple greedy decoding does not (Collobert et al., 2011). Therefore, in a bidirectional LSTM (biLSTM) model with a CRF layer, illegal transitions are rare compared to models that select the best scoring tag for each token.

Due to the high variance observed in the performance of NER models (Reimers and Gurevych, 2017) it is important to have fast training times to allow for multiple runs of these models. However, as the CRF forward algorithm is $O(NT^2)$, where N is the length of the sentence and T is the number of possible tags, it slows down the training significantly. Moreover, substantial effort is required to build an optimized, correct implementation of this layer. Alternately, training with a cross-entropy loss runs in $O(N)$ for sparse labels and popular deep learning toolkits provide an easy to use, parallel version of this loss which brings the runtime down to $O(\log N)$.

We believe that, due to the strong contextualized local features with infinite context created by today's neural models, global features used in the CRF do little more than enforce the rules of an encoding scheme. Instead of traditional CRF training, we propose training with a cross-entropy loss and using Viterbi decoding (Forney, 1973) with heuristically determined transition probabilities that prohibit illegal transitions. We call this constrained

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 1841–1848

decoding and find that it allows us to train models in half the time while yielding F1 scores comparable to CRFs.

2 Method

Training a tagger with a CRF is normally done by minimizing the negative log likelihood of the sequence of gold tags given the input, parameterized by the model, where the probability of the sequence is given by

$$P(y|x;\theta) = \frac{e^{\sum_i \sum_j w_j f_j(y_{i-1},y_i,x,i))}}{\sum_{y' \in Y} e^{\sum_i \sum_j w_j f_j(y'_{i-1},y'_i,x,i)}}$$

By creating a feature function, f_j, that is span-encoding-scheme-aware, we can introduce constraints that penalize any sequence that includes an illegal transition by returning a large negative value. Note the summation over all possible tag sequences. While efficient dynamic programs exist to make this sum tractable for linear-chain CRFs with Markov assumptions, this is still a costly normalization factor to compute.

In neural models, these feature functions are represented as a transition matrix that represents the score of moving from one tag y at index i to another at $i+1$. We implement a mask that effectively eliminates invalid IOBES transitions by setting those scores to large negative values. By applying this mask to the transition matrix we can simulate feature functions that down-weigh illegal transitions.

Contrast the CRF loss with the token-level cross-entropy loss where y is the correct labels and $\hat{y}$ is the model's predictions.

$$L_{\text{cross-entropy}} = -\sum_i y_i \log(\hat{y}_i)$$

Here we can see that the loss for each element in the input i can be computed independently due to the lack of a global normalization factor. This lack of a global view is potentially harmful, as we lose the ability to condition on the previous label decision to avoid making illegal transitions. We hypothesize that, using our illegal transition heuristics, we can create feature functions that do not have to be trained, but can be applied at test time and allow for contextual coherence while using a cross-entropy loss.

We can use the mask directly as the transition matrix to calculate the maximum probability sequence while avoiding illegal transitions for models that were not trained with a CRF. Using these transitions scores in conjunction with cross-entropy trained models, we can achieve comparable models that train more quickly. We call this method constrained decoding.

Constrained decoding is relatively easy to implement, given a working CRF implementation, all one needs to do is apply the transition mask to the CRF transition parameters to create a constrained CRF. Replacing the transition parameters with the mask yields our constrained decoding model. Starting from scratch, one only needs to implement Viterbi decoding, using the mask as transition parameters, to implement the constrained decoding model—avoiding the need for the CRF forward algorithm and the CRF loss.

For constrained decoding, we leverage the IOBES tagging scheme rather than BIO tagging, allowing us to inject more structure into the decoding mask. Early experiments with BIO tagging failed to show the large gains we realized using IOBES tagging for the reasons mentioned in Section 4.

3 Experiments & Results

To test if we can replace the CRF with constrained decoding we use two sequential prediction tasks: NER (CoNLL 2003 (Tjong Kim Sang and De Meulder, 2003), WNUT-17 (Derczynski et al., 2017), and OntoNotes (Hovy et al., 2006)) and slot-filling (Snips (Coucke et al., 2018)). For each (task, dataset) pair we use common embeddings and hyperparameters from the literature. The baseline models are biLSTM-CRFs with character compositional features based on convolutional neural networks (Dos Santos and Zadrozny, 2014) and our models are identical except we train with a cross-entropy loss and use the encoding scheme constraints as transition probabilities instead of learning them with a CRF. Our hyper-parameters mostly follow Ma and Hovy (2016), except we use multiple pre-trained word embeddings concatenated together (Lester et al., 2020). For Ontonotes we follow Chiu and Nichols (2016). See Section A.7 or the configuration files in our implementation for more details.

As seen in Table 1, in three out of four datasets constrained decoding performs comparably or better than the CRF in terms of F1. OntoNotes is

Dataset	Model	mean	std	max
CoNLL	CRF	**91.61**	0.25	**92.00**
	Constrain	91.44	0.23	91.90
WNUT-17	CRF	40.33	1.13	**41.99**
	Constrain	**40.59**	1.06	41.71
Snips	CRF	96.04	0.28	**96.35**
	Constrain	**96.07**	0.17	96.29
OntoNotes	CRF	**87.43**	0.26	**87.57**
	Constrain	86.13	0.17	86.72

Table 1: Tagging results on a variety of datasets. The CRF model is a standard biLSTM-CRF while the Constrain model is a biLSTM trained with a cross-entropy loss that uses heuristic transition scores, created from the illegal transitions, for test time decoding. OntoNotes is the only dataset where the difference in performance between the CRF and constrained decdoing is statistically significant ($p < 0.5$). All scores are entity-level F1 and are reported across 10 runs.

the only dataset with a statistically significant difference in performance. We explore this discrepancy in Section 4. Similarly, Table 2 shows that when we apply constrained decoding to a variety of internal datasets, which span a diverse set of specific domains, we do not observe any statistically significant differences in F1 between CRF and constrained decoding models.

The models were trained using Mead-Baseline (Pressel et al., 2018), an open-source framework for creating, training, evaluating and deploying models for NLP. The constrained decoding tagger performs much faster at training time. Even when compared to the optimized, batched CRF provided by Mead-Baseline, it trained in 51.2% of the time as the CRF.

In addition to faster training times, training our constrained models produces only 65% of the CO_2 emissions that the CRF does. While GPU computations for the constrained model draw 1.3 times more power—due to the greater degree of possible parallelism in the cross-entropy loss function—than the CRF, the reduction in training time results in smaller carbon emissions as calculated in Strubell et al. (2019).

Constrained decoding can also be applied to a CRF. The CRF does not always learn the rules of a transition scheme, especially in early training iterations. Applying the constraints to the CRF can improve both F1 and convergence speed. We establish this by training biLSTM-CRF models with and without constraints on CoNLL 2003. We find that the constraint mask yields a small (albeit statistically insignificant) boost in F1 as shown in Table 4.

Task	Domain	Δ
NER	Generic NER	0.80
Slot Filling	Customer Service	0.21
	Automotive	-0.68
	Cyber Security	0.84

Table 2: Entity-level F1 comparing a constrained CRF model with a constrained decoding model. Due to the nature of the the data we present the relative performance difference between the two models. We see some improvements and some drops in performance but, once again, there is not a statistically significant difference between the CRF and constrained decoding.

Task	Dataset	Δ
NER	CoNLL	-0.03
	WNUT-17	0.65
	OntoNotes	-1.48
	Snips	0.03

Table 3: Results on well-known datatsets presented as relative differences to help frame results in Table 2

Our experiments suggest that injecting prior knowledge of the transition scheme helps the model to focus on learning the features for sequence tagging tasks (and not the transition rules themselves) and train faster. Table 5 shows that our constrained model converged [1] on CoNLL 2003 faster on average than an unconstrained CRF.

4 Analysis

The relatively poor performance of constrained decoding on OntoNotes suggests that there are several classes of transition that it cannot model. For example, the transition distribution between entity types,

[1] We define convergence as the epoch where development set performance stops improving

Model	mean	std	max
Unconstrained	91.55	0.26	91.79
Constrained	**91.61**	0.25	**92.00**

Table 4: Results of biLSTM-CRF models with and without constraints evaluated with entity-level F1 on the CoNLL 2003 dataset. Scores are reported across 10 runs. We see that while, in theory, the CRF should learn the constraints, injecting this knowledge gives a gain in performance.

Model	mean	std	min	max
Unconstrained	72.4	21.0	16	97
Constrained	**60.6**	23.3	37	89

Table 5: Using the constraints while training a biLSTM-CRF tagger on the CoNLL dataset result in a statistically significant ($p < 0.5$) decrease in the number of epochs until convergence. Scores are reported across 30 runs.

or the prior distribution of entities. We analyzed the datasets to identify the characteristics that cause constrained decoding to fail.

One such presumably obvious characteristic is the number of entity types. However, our experiments suggest that number of entity types does not affect performance: Snips has more entity types than OntoNotes yet constrained decoding works better for Snips.

We define an ambiguous token as a token whose type has multiple tag values in the dataset. For example the token "Chicago" could be "I-LOC" or "I-ORG" in the phrases "the Chicago River" and "the Chicago Bears" respectively. Such ambiguous tokens are the ones for which we expect global features to be particularly useful. A "strictly dominated token" is defined as a token that can only take on a single value due to the legality of the transition from the previous tag. In the above example given that "the" was a "B-LOC" then "Chicago" is strictly dominated and forced to be an "I-LOC'. Contrast this with a non-strictly dominated token that can still have multiple possible tag values when conditioned on the previous tag. As constrained decoding eliminates illegal transitions we would expect that it would perform well on datasets where a large proportion of ambiguous tokens are strictly dominated. This tends to hold true—only 15.9% of OntoNotes' ambiguous tokens are strictly dominated while 70.7% of CoNLL's tokens are and for WNUT-17 73.6% are.

We believe that the ambiguity of the first and last token of an entity also plays a role. Once we start an entity, constrained decoding vastly narrows the scope of decisions that need to be made. Instead of making a decision over the entire set of tags, we only decide if we should continue the entity with an "I-" or end it with an "E-". Therefore, we expect constrained decoding to work well with datasets that have fairly unambiguous entity starts and ends. We quantify this by finding the proportion of entities that begin (or end) with an unambiguous type, that is, the first token of an entity only has a single label throughout the dataset, for example, "Kuwait" is only labeled with "S-LOC" in the CoNLL dataset. We call these metrics "Easy First" and "Easy Last" respectively and find that datasets with higher constrained decoding performance also have a higher percentages of entities with an easy first or last token. A summary of these characteristics for each dataset is found in Table 6.

This also explains why constrained decoding doesn't work as well for BIO-encoded CoNLL as it does for IOBES. When using the IOBES format, more tokens are strictly dominated. The other stark difference is the proportion of "Easy Last" entities. Without the "E-" token, much less structure can be injected into the model, resulting in decreased performance of constrained decoding. These trends also hold true in internal datasets, where the Automotive dataset had the fewest incidences of each of these phenomena.

While not perfect predictors for the performance of constrained decoding, the metrics chosen are good proxies and can be used as a prescriptive measure for new datasets.

5 Previous Work

Our approach is similar in spirit to previous work in NLP where constraints are introduced during training and inference time (Roth and Yih, 2005; Punyakanok et al., 2005) to lighten the computational load, and to Strubell et al. (2018) where prior knowledge is injected into the model by manual manipulation. In our approach, however, we focus specifically on manipulating the model weights themselves rather than model features.

There have been attempts to eliminate the CRF layer, notably, Shen et al. (2017) found that an additional LSTM greedy decoder layer is competitive with the CRF layer, though their baseline is much weaker than the models found in other work. Additionally, their decoder has an auto-regressive relationship that is difficult to parallelize and, in practice, there is still significant overhead at training time. Chiu and Nichols (2016) mention good results with a similar technique but don't provide in-depth analysis, metrics, or test its generality.

6 Conclusion

For sequence tagging tasks, a CRF layer introduces substantial computational cost. We propose replacing it with a lightweight technique, constrained

Dataset	Tag Types	Ambiguity	Strictly Dominated	Easy First	Easy Last
CoNLL (IOBES)	4	8.8%	71.2%	58.3%	94.0%
CoNLL (BIO)	4	7.4%	59.6%	68.5%	57.4%
WNUT-17	6	3.6%	74.3%	82.9%	97.0%
OntoNotes	18	14.9%	15.9%	16.2%	55.9%
Snips	39	24.5%	26.7%	32.4%	91.1%

Table 6: Analysis of the tag dynamics and co-occurrence. We see that OntoNotes is an outlier in the percentage of ambiguous tokens that are strictly dominated by their context, the entities that have easy to spot starting tokens, and entities with clearly defined ends. All of these quirks of the data help explain why we only see a statistically significant performance drop for OntoNotes.

decoding, which doubles the speed of training with comparable F1 performance. We analyze the algorithm to understand where it might work or fail and propose prescriptive measures for using it.

The broad theme of the work is to find simple and computationally efficient modifications of current networks and suggest possible failure cases. While larger models have shown significant improvements, we believe there is still relevance in investigating small, targeted changes. In the future, we want to explore similar techniques in other common NLP tasks.

References

Jason P.C. Chiu and Eric Nichols. 2016. Named Entity Recognition with Bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibault Lavril, Maël Primet, and Joseph Dureau. 2018. Snips Voice Platform: an Embedded Spoken Language Understanding System for Private-by-design Voice Interfaces. *arXiv preprint*, arXiv:1805.10190.

Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the WNUT2017 Shared Task on Novel and Emerging Entity Recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 140–147, Copenhagen, Denmark. Association for Computational Linguistics.

Cícero Nogueira Dos Santos and Bianca Zadrozny. 2014. Learning Character-level Representations for Part-of-speech Tagging. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1818–II–1826. JMLR.org.

G.D. Forney. 1973. The Viterbi Algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90% Solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, NAACL-Short '06, pages 57–60, Stroudsburg, PA, USA. Association for Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 260–270.

Brian Lester, Daniel Pressel, Amy Hemmeter, Sagnik Ray Choudhury, and Srinivas Bangalore. 2020. Multiple Word Embeddings for Increased Diversity of Representation. *arXiv preprint arXiv:2009.14394*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word

Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Daniel Pressel, Sagnik Ray Choudhury, Brian Lester, Yanjie Zhao, and Matt Barta. 2018. Baseline: A Library for Rapid Modeling, Experimentation and Development of Deep Learning Algorithms Targeting NLP. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 34–40. Association for Computational Linguistics.

Vasin Punyakanok, Dan Roth, Wen-tau Yih, and Dav Zimak. 2005. Learning and Inference over Constrained Output. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, pages 1124–1129, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Lev Ratinov and Dan Roth. 2009. Design Challenges and Misconceptions in Named Entity Recognition. In *CoNLL 2009 - Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155.

Nils Reimers and Iryna Gurevych. 2017. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348, Copenhagen, Denmark. Association for Computational Linguistics.

Dan Roth and Wen-tau Yih. 2005. Integer Linear Programming Inference for Conditional Random Fields. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 736–743, New York, NY, USA. ACM.

Yanyao Shen, Hyokun Yun, Zachary Lipton, Yakov Kronrod, and Animashree Anandkumar. 2017. Deep Active Learning for Named Entity Recognition. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 252–256, Vancouver, Canada. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and Policy Considerations for Deep Learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-Informed Self-Attention for Semantic Role Labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038, Brussels, Belgium. Association for Computational Linguistics.

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 - Volume 4*, CONLL '03, pages 142–147, Stroudsburg, PA, USA. Association for Computational Linguistics.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272.

A Reproducibility

A.1 Hyperparameters

Mead/Baseline is a configuration file driven model training framework. All hyperparameters are fully specified in the configuration files included with the source code for our experiments.

A.2 Statistical Significance

For all claims of statistical significance we use a t-test as implemented in `scipy` (Virtanen et al., 2020) and using an alpha value of 0.05.

A.3 Computational Resources

All models were trained on a single NVIDIA 1080Ti. While multiple GPUs were used for training many models in parallel to facilitate testing many datasets and to estimate the variability of the method, the actual model can easily be trained on a single GPU.

A.4 Evaluation

To calculate metrics, entity-level F1 is used for NER and slot-filling. In entity-level F1, entities are created from the token-level labels and compared to the gold entities. Entities that match on both type and boundaries are considered correct while a mismatch in either causes an error. The F1 score is then calculated using these entities. We use the

Dataset	Model	Parameters
CoNLL	CRF	4,658,190
	Constrain	4,657,790
	Unconstrained CRF	4,658,190
WNUT-17	CRF	12,090,032
	Constrain	12,089,248
Snips	CRF	5,940,866
	Constrain	5,924,737
OntoNotes	CRF	12,090,032
	Constrain	12,089,248

Table 7: The number of parameters for different models.

evaluation code that ships with the framework we use, MEAD/Baseline, which we have bundled with the source code for our experiments.

A.5 Model Size

The number of parameters in different models can be found in Table 7.

A.6 Dataset Information

Relevant information about datasets can be found in Table 8. The majority of data is used as distributed, except we convert NER and slot-filling datasets to the IOBES format. All public datasets are included in the supplementary material. A quick overview of each dataset follows:

CoNLL: A NER dataset based on news text. We converted the IOB labels into the IOBES format. There are 4 entity types, `MISC`, `LOC`, `PER`, and `LOC`.

WNUT-17: A NER dataset of new and emerging entities based on noisy user text. We converted the BIO labels into the IOBES format. There are 6 entity types, `corporation`, `creative-work`, `group`, `location`, `person`, and `product`.

OntoNotes: A much larger NER dataset. We converted the labels into the IOBES format. There are 18 entity types, `CARDINAL`, `DATE`, `EVENT`, `FAC`, `GPE`, `LANGUAGE`, `LAW`, `LOC`, `MONEY`, `NORP`, `ORDINAL`, `ORG`, `PERCENT`, `PERSON`, `PRODUCT`, `QUANTITY`, `TIME`, and `WORK_OF_ART`.

Snips: A slot-filling dataset focusing on commands one would give a virtual assistant. We converted the dataset from its normal format of two associated files, one containing surface terms and one containing labels in the more standard CoNLL file format and converted the labels into the IOBES format. There are 39 entity types, `album`, `artist`, `best_rating`, `city`, `condition_description`, `condition_temperature`, `country`, `cuisine`, `current_location`, `entity_name`, `facility`, `genre`, `geographic_poi`, `location_name`, `movie_name`, `movie_type`, `music_item`, `object_location_type`, `object_name`, `object_part_of_series_type`, `object_select`, `object_type`, `party_size_description`, `party_size_number`, `playlist`, `playlist_owner`, `poi`, `rating_unit`, `rating_value`, `restaurant_name`, `restaurant_type`, `served_dish`, `service`, `sort`, `spatial_relation`, `state`, `timeRange`, `track`, and `year`.

A.7 Hyper Parameters

Table 9 details the various hyper-parameters used to train models for each dataset. For all datasets the only difference between the baseline CRF model and the model using constrained decoding is that the CRF has learnable transition parameters in the final layer while the constrained decoding model sets these transitions parameters manually based on the rules of the span encoding scheme. The framework we use, Mead-Baseline, is configuration file driven and we have included the configuration files used on our experiments in the supplementary material.

Dataset		Train	Dev	Test	Total
CoNLL	Examples	14,987	3,466	3674	22137
	Tokens	204,567	51,578	46,666	302,811
WNUT-17	Examples	3,394	1,009	1,287	5,690
	Tokens	62,730	15,733	23,394	101,857
OntoNotes	Examples	59,924	8,528	8,262	76,714
	Tokens	1,088,503	147,724	152,728	1,388,955
Snips	Examples	13,084	700	700	14,484
	Tokens	117,700	6,384	6,354	130,438

Table 8: Example and token count statistics for public datasets used.

HyperParameter	CoNLL	Ontonotes	Snips	WNUT-17
Embedding	6B + Senna	6B + Senna	6B + GN	27B + w2v-30M + 840B
Character Filter Size	3	3	3	3
Character Feature Size	30	30	30	30
Character Embed Size	30	20	30	30
RNN Type	biLSTM	biLSTM	biLSTM	biLSTM
RNN Size	400	400	400	200
RNN Layers	1	2	1	1
Drop In	0.1	0.1	0.1	0.0
Drop Out	0.5	0.63	0.5	0.5
Batch Size	10	9	10	20
Epochs	100	100	100	60
Learning Rate	0.015	0.008	0.015	0.008
Momentum	0.9	0.9	0.9	0.9
Gradient Clipping	5.0	5.0	5.0	5.0
Optimizer	SGD	SGD	SGD	SGD
Patience	40	40	40	20
Early Stopping Metric	f1	f1	f1	f1
Span Type	IOBES	IOBES	IOBES	IOBES

Table 9: Hyper-parameters used for each dataset. "Embedding" is the type of pre-trained word embeddings used. 6B, 27B, and 840B are GloVe embeddings (Pennington et al., 2014) with 27B having been trained on Twitter, Senna is embeddings from Collobert et al. (2011), GN is vectors trained on Google News with word2vec from Mikolov et al. (2013) and w2v-30M are word2vec vectors trained on Twitter from Pressel et al. (2018). "Character Filter Size" is the number of token the character compositional convolutional neural network cover is a single window, "Character Feature Size" is the number of convolutional features maps used, and "Character Embed Size" is the dimensionality of the vectors each character is mapped to before it is the input to the convolutional network. The "RNN Size" is the size of the output after the RNN which means that bidirectional RNNs are composed to two RNNs, one in each direction, where both are half the "RNN Size". "Drop In" is the probability that an entire token will be drop out from the input, while "Drop Out" is the probability that individual neurons are dropped out (Srivastava et al., 2014).