Named Entity Recognition as Structured Span Prediction

Urchade Zaratiana[*†], **Nadi Tomeh**[†], **Pierre Holat**[*†], **Thierry Charnois**[†]
[*] FI Group, [†] LIPN, CNRS UMR 7030, France
{urchade.zaratiana,pierre.holah}@fi-group.com
{charnois,tomeh}@lipn.fr

Abstract

Named Entity Recognition (NER) is an important task in Natural Language Processing with applications in many domains. While the dominant paradigm of NER is sequence labelling, span-based approaches have become very popular in recent times but are less well understood. In this work, we study different aspects of span-based NER, namely the span representation, learning strategy, and decoding algorithms to avoid span overlap. We also propose an exact algorithm that efficiently finds the set of non-overlapping spans that maximizes a global score, given a list of candidate spans. We performed our study on three benchmark NER datasets from different domains. We make our code publicly available at https://github.com/urchade/span-structured-prediction.

1 Introduction

Named Entity Recognition (NER) is an important task in natural language processing whose goal is to identify and extract salient entities such as persons, organizations and locations from texts. NER systems are typically designed as sequence labelling: token-level prediction utilizing the BIO scheme. While traditional approaches use hand-crafted features along with classical Machine Learning algorithms such as SVMs or decision trees (Carreras et al., 2002; Li et al., 2004), deep learning models learn features directly from the data using for example bi-directional LSTMs (Huang et al., 2015; Lample et al., 2016; Akbik et al., 2018) or more recently pre-trained language models such as BERT (Devlin et al., 2019; Yu et al., 2020).

Recently, span-based NER has gained in popularity. Unlike sequence tagging which operates at the token level, span-based NER operates directly at the span level. The main idea is to enumerate all possible contiguous sequence of tokens of an input text and predict their identity (Lee et al., 2017). One of the major advantages of the span-based NER is that it can learn a rich representation of the span instead of only learning the representation of each token. In addition, a recent study by Fu et al. (2021) reveals that span-based NERs are better in a context with more OOV words and Li et al. (2021) showed that span-based NERs are much better than sequence labelling in settings with unlabelled entities (missing entities due to annotation errors).

However, unlike sequence labelling, *unconstrained* span-based approaches tend to produce overlapping entities, which is undesirable for flat, non-overlapping NER tasks. To avoid overlap in span-based NER, two main approaches have been adopted in the literature. The first is the Semi-Markov conditional random field (Sarawagi and Cohen, 2005) that trains a globally normalized model and then uses a Viterbi algorithm to produce the optimal segmentation without span overlap, we call this approach *Semi-CRF*. The second algorithm is the one employed by Li et al. (2021) for locally normalized span-based NER; it first eliminates all non-entity spans and deals with the overlap conflict by keeping the span with the highest prediction probability while eliminating the others. In this work, we call this approach *greedy decoding*.

In this paper, we analyze and compare two formulations of span-based NER. The first is a *segmentation* model of the Semi-CRF; the second is the two-step pipeline of span filtering and decoding. In addition to greedy decoding, we propose an exact algorithm based on Maximum Weighted Independent Set (MWIS) (Hsiao et al., 1992; Pal and Bhattacharjee, 1996) on internal graphs. We build such graphs to encode the overlapping structure between spans. This formulation of the NER task is novel up to our knowledge. For completeness, we include in the comparison a token-based sequence labeling model with a linear-chain CRF.

In order to understand the effect of span representation, we explore different alternatives includ-

Proceedings of the Workshop on Unimodal and Multimodal Induction of Linguistic Structures (UM-IoS), pages 1 - 10

ing max-pooling, convolution and endpoints (representing span by its extreme tokens) and show that endpoints are effective across models and datasets.

Our contributions can be summarized as follow:

- We propose an exact decoding algorithm to eliminate span overlap on locally trained models that overcomes the myopic bias of the greedy approach (Li et al., 2021). We present a detailed comparison with global models.
- We investigate different span representations for span-based NER when using pretrained Transformer models. Our experiment provide a confirmation that the endpoint representation, the currently dominant representation strategy is the most robust.
- We conduct few-shot performance analysis for different modelling. We found that classical sequence labeling models provide strong result for datasets with few entity types, while span-based approaches are better for larger type sets.

Our code for models and experiments is publicly available.[1]

2 Span Representation

Given an input sequence $\boldsymbol{x} = [x_1, \ldots, x_n]$, a span (i, j) is the contiguous segment of tokens $[x_i, \ldots, x_j]$. The goal of representation is to compute an embedding vector for each span of an input text which can be used for downstream prediction tasks. We denote $\boldsymbol{h}_i \in \mathbb{R}^{d_h}$ the representation of the word at the position i and $\boldsymbol{s}_{ij} \in \mathbb{R}^{d_s}$ the representation of the span (i, j) with the width $k = j - i + 1$; here $d_h, d_s \in \mathbb{N}^+$ are respectively the embedding sizes for word and span representations. The token representations are computed using a BERT-based model (Devlin et al., 2019). However, since BERT-based tokenization divides the input words into subwords, we take the first subword to represent the whole word, which has proven to be very competitive for several token classification tasks (Beltagy et al., 2019). In the following, we present different approaches for representing the spans.

Endpoints This representation consists in representing a span using the representation of the tokens of its right and left extremities, in addition to a span width feature. Specifically, the representation of the span (i, j), $\boldsymbol{s}_{ij}$ is computed as:

$$\boldsymbol{s}_{ij} := [\boldsymbol{h}_i; \boldsymbol{h}_j; \boldsymbol{w}_k] \quad (1)$$

where $\boldsymbol{w}_k$ is a learned vector of width k and $[;]$ denotes the concatenation operation. Endpoints have been widely used in previous works for span prediction tasks such as NER and coreference resolution (Lee et al., 2017; Luan et al., 2019; Zhong and Chen, 2021).

Span representation	Num params.
Endpoints	$(2d_h + d_k)C$
Maxpool	$d_h C$
Convolution	$\frac{1}{2}d_h^2 K(K+1) + d_h C$
Convolution (shared)	$d_h^2 K + d_h C$
FirstToken	$d_h^2 K + d_h C$

Table 1: Number of parameters for different representation, without including the word representation layer which is the same for any approach. d_h, K and C are respectively the word embedding size, the maximum span width and the number of classes. Blue terms are parameters for computing span representations and Red terms denote number of parameters for the final layer.

Max-pooling Since spans consist of a contiguous segment of tokens, pooling operations are a fairly natural way to compute their representations. In this context, we use an element-wise max-pooling operation to all tokens inside the span. Formally,

$$\boldsymbol{s}_{ij} := \texttt{MAX}([\boldsymbol{h}_i; \boldsymbol{h}_{i+1}; \ldots; \boldsymbol{h}_j]) \quad (2)$$

where MAX is the element-wise max pooling operation. Max-pooling has been previously used by Eberts and Ulges (2020) for joint entity and relation extraction.

Convolution Instead of simply applying the pooling operation, we explored aggregating tokens using learned filters via convolution. Specifically, representations of all spans of size k are computed simultaneously using a 1D convolution of kernel size k. To keep the number of parameters linear with respect to the maximum span width, we share the convolution weights across the different span widths.

$$\boldsymbol{s}_{ij} := \texttt{Conv1D}_k([\boldsymbol{h}_i; \boldsymbol{h}_{i+1}; \ldots; \boldsymbol{h}_j]) \quad (3)$$

Lei et al. (2021) used this convolutional approach to represent spans for keyphrase extraction.

[1] Anonymized for review.

FirstToken For this representation, we only use the start token along with span width information:

$$\boldsymbol{s}_{ij} := \boldsymbol{W}^{(k)}\boldsymbol{h}_i \tag{4}$$

where $\boldsymbol{W}^{(k)} \in \mathbb{R}^{d_h \times d_h}$ is the weight matrix associated with width k. Note that the computation of the representation of all spans for this approach can be done in parallel and in a single line of code using einsum operation (Rogozhnikov, 2022). This representation was inspired by the synthetic attention from Tay et al. (2021), where the authors predict attention scores without pairwise interaction.

Number of parameters The number of parameters required for each span representation is shown in Table 1.

3 Span scores

We model the task of NER as assigning to each span (i, j) a label from a set of C different types that correspond to named-entity types and special `null` type, indicating that the span does not correspond to an entity. Label assignment is constrained so that no pair of overlapping spans have entity types (both different from `null`).

We present two models to solve this structured prediction problem: a locally normalized approach with a zero-order scoring function which does not take into consideration the interactions between label assignment (§4); and a globally normalized approach with first-order scoring function which considers dependencies between pairs of consecutive spans (§5).

Both formulations employ the following span scoring function. Given a span representation $\boldsymbol{s}_{ij}$, the logits $\boldsymbol{\phi}(i, j) \in \mathbb{R}^C$ for the C different labels are computed using a non-linear activation function followed by an affine transformation:

$$\boldsymbol{\phi}(i, j) = \boldsymbol{W}\texttt{ReLU}(\boldsymbol{s}_{ij}) + \mathbf{f} \tag{5}$$

where $\boldsymbol{W} \in \mathbb{R}^{d_s \times C}$ is the final weight matrix, $\mathbf{f} \in \mathbb{R}^C$ is the bias vector, and `ReLU` is the activation function. We denote by $\boldsymbol{\phi}(i, j, l) \in \mathbb{R}$ the (unnormalized) score of the label l for the span (i, j).

4 Locally Normalized Models

Under this approach, we perform span labeling in two steps, span classification followed by a decoding step.

4.1 Span Classification

Each span (i, j) is assigned its highest scoring label $\hat{l}_{ij} = \arg\max_l \boldsymbol{\phi}(i, j, l)$, and we denote $\hat{k}_{ij}$ the corresponding highest score. The set of spans classified as entities may contains overlapping spans, a decoding step is therefore required to select a subset with no overlaps.

We learn the parameters[2] of this classifier under a locally normalized setup. The training's objective is to maximize the likelihood for every span label (up to a maximum lenght K) from the training data. The loss function is as follows:

$$\mathcal{L} = -\sum_{(i,j,l)\in\mathcal{T}} \log \frac{\exp\{\phi(i, j, l)\}}{\sum_{l'} \exp\{\phi(i, j, l')\}} \tag{6}$$

which is the well-known cross-entropy loss.

4.2 Greedy Decoding

Let $S = \{(i, j) : \hat{l}_{ij} \neq \texttt{null}\}$ be the set of spans classified as entities. The goal of decoding is to find the subset of S that maximizes a global score function:

$$E^* = \arg\max_{E\subseteq S} \sum_{(i,j)\in E} \hat{k}_{ij} \tag{7}$$
$$\text{s.t. } \forall e, e' \in E : \texttt{!overlap}(e, e')$$
$$\forall u \notin E, \exists e \in E : \texttt{overlap}(e, u)$$

where $\texttt{overlap}(e, e')$ is True if the spans e and e' overlap but are not equal. The first constraint in Eq. 7 ensures that the set E is *independent*, i.e. it doesn't contains overlapping spans; the second constraint ensures that it is *maximal*, i.e. adding any other span breaks the no-overlap constraint.

Greedy decoding constructs an approximation to E^* by iteratively adding the highest-scoring entity not overlapping with any previously selected entity. This algorithm is efficient and has a complexity of $O(n \log n)$ with $n = |S|$.

4.3 Exact Decoding with MWIS

We define an *overlapping graph* as the graph G whose nodes are the elements of S and contains an edge between each pair of overlapping spans. Its adjacency matrix is defined as:

$$\boldsymbol{A}[e, e'] = \begin{cases} 1, & \text{if } \texttt{overlap}(e, e') \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

[2]The parameters include all weight matrices from span representation and scoring functions. We omit the parameters from the notation for simplicity.

We associate a weight to each node as provided by its label score $\boldsymbol{\phi}(i, j, \hat{l}_{ij})$.

An exact solution to Eq. 7 is given by the Maximum Weight Independent Set (MWIS) of the overlapping graph. For general graphs, computing the MWIS is NP-Hard but since our graph can be seen as an *interval graph* (spans can be considered as intervals over their start and end positions), MWIS has a complexity of $O(n \log n)$ or $O(n)$ if the spans are sorted by their endpoint (Hsiao et al., 1992).

4.4 Exhaustive Search Decoding

For efficient decoding, the scoring function in Eq. 7 decomposes as a sum over graph nodes. More complex scoring functions do not necessarily admit efficient decoding. Finding an optimal set under the mean scoring function for instance, that is $\frac{1}{|E|}\sum_{(i,j)\in E}\hat{k}_{ij}$, requires enumerating all possible candidates subsets of S, which is NP-Hard (Johnson et al., 1988; Raman et al., 2007) but feasible for reasonably small interval graphs. In this paper, we experiment with this scoring functions but leave more complex ones for future work.

5 Globally normalized model

Under this approach, NER is modeled using a semi-Markov segmentation CRF introduced by Sarawagi and Cohen (2005). The input sentence $\boldsymbol{x}$ is segmented into a labeled sequence of spans $\boldsymbol{y}$. Each segmentation is scored as:[3]

$$\Omega(\boldsymbol{y}) = \sum_{y_k=(i,j,l)} \boldsymbol{\phi}(i, j, l) + \boldsymbol{T}_{l',l} \qquad (9)$$

with $y_k = (i, j, l)$ being the labeled span at position k. Unlike the scoring function in Eq. 7, the score here contains the transition scores from label l' at position $k-1$ to label l, in the *learnable* matrix $\boldsymbol{T}$.

Training The parameters of the model are learned to maximize the conditional probability of the gold segmentation in the training data. The probability of a segmentation is computed by globally normalizing the score: $P(\boldsymbol{y}|\boldsymbol{x}) = \exp\{\Omega(\boldsymbol{y}) - Z\}$, where Z is the log partition function $\log \sum_{\boldsymbol{y}\in\mathcal{Y}(\boldsymbol{x})} \exp\{\Omega(\boldsymbol{y})\}$, which sums over all possible segmentation $\mathcal{Y}(\boldsymbol{x})$. This normalization term can be computed in polynomial time using dynamic programming.

[3]We drop the dependence on the input $\boldsymbol{x}$ for simplicity.

Decoding algorithm	Time complexity
CRF	$O(L\|Y\|^2)$
Semi-CRF	$O(LK\|Y\|^2)$
Greedy decoding	$O(n \log n)$
MWIS	$O(n \log n)$
Exhaustive Search (EXT)	$O(3^{n/3})$

Table 2: This table reports the complexity of the different decoding algorithms. L is the input length, K the maximum segment width, $|Y|$ the number of classes and n the number of spans after filtering non-entities, which is approximately equal to $0.15 \times L$ empirically.

Following (Sarawagi and Cohen, 2005), we assume that segments have strictly positive lengths, adjacent segments touch and we assume that non-entity spans have unit length. For instance, a segmentation of the sentence "Michael Jordan eats an apple ." would be *Y=[(0, 1, PER), (2, 2, O), (3, 3, O), (4, 4, O), (5, 5, O)]*.

Decoding Selecting the most probable segmentation $\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}\in\mathcal{Y}(\boldsymbol{x})} \Omega(\boldsymbol{y})$ is efficiently performed using the segmental variant of the Viterbi algorithm (Sarawagi and Cohen, 2005).

6 Experimental Setup

6.1 Datasets

We evaluated our model on three benchmark datasets for Named Entity Recognition: Conll-2003 (Tjong Kim Sang and De Meulder, 2003), OntoNotes 5.0 (Weischedel et al., 2013) and TDM (Hou et al., 2021). Conll-2003 is a dataset from the news domain that was designed for extracting entities such as Person, Location and Organisation. OntoNotes 5.0 is a large corpus comprising various genres of text including newswire, broadcast news and telephone conversation. It contains in total 18 different entity types such as Person, Organization, Location, Product or Date. TDM is a NER dataset that was recently published and it was designed for extracting Tasks, Datasets, and Metrics entities from Natural Language Processing papers.

Dataset	**Entity types**	**Train / Dev / Test**
Conll-2003	4	14987 / 3466 / 3684
OntoNotes 5.0	18	48788 / 7477 / 5013
TDM	3	1000 / 500 / 500

Table 3: Dataset statistics

Model	Span Representation											
	Convolution			*Endpoints*			*Maxpool*			*FirstToken*		
	P	R	F	P	R	F	P	R	F	P	R	F
Conll-2003												
Local	91.40	89.86	90.62	91.07	90.48	90.77	90.52	90.52	90.52	90.34	89.25	89.79
+ Greedy	91.97	89.74	90.84	91.5	90.1	90.79	91.26	90.1	90.67	90.64	89.08	89.85
+ MWIS	91.97	89.76	**90.85**	91.5	90.11	90.8	91.22	90.09	90.65	90.64	89.08	89.85
+ EXT	91.97	89.75	**90.85**	91.54	90.11	**90.82**	91.33	90.11	**90.71**	90.66	89.09	**89.86**
Semi-CRF	89.45	88.99	89.22	89.64	89.23	89.43	89.48	88.82	89.15	89.5	89.17	89.33
OntoNotes 5.0												
Local	88.59	88.99	88.79	88.06	89.55	88.8	88.42	89.34	88.88	88.18	88.73	88.45
+ Greedy	89.3	88.62	**88.96**	88.93	89.0	88.96	89.38	88.83	89.11	88.8	88.22	88.51
+ MWIS	89.26	88.61	88.93	88.9	88.98	88.94	89.38	88.87	**89.13**	88.81	88.26	**88.53**
+ EXT	89.31	88.61	88.95	88.95	89.01	**88.98**	89.37	88.79	89.08	88.80	88.20	88.50
Semi-CRF	87.35	87.76	87.55	87.36	88.26	87.81	87.04	87.99	87.51	87.11	87.86	87.48
TDM												
Local	73.05	69.38	71.15	67.75	69.88	68.78	70.86	70.69	70.73	68.54	65.06	66.74
+ Greedy	75.86	68.28	**71.84**	75.12	67.82	71.26	73.24	69.43	71.26	69.82	64.40	66.99
+ MWIS	75.46	68.07	71.55	75.25	68.12	**71.48**	73.31	69.53	**71.34**	69.89	64.50	67.07
+ EXT	75.72	68.07	71.67	74.63	66.97	70.57	73.24	69.43	71.26	69.82	64.40	66.99
Semi-CRF	68.34	72.55	70.35	69.38	72.85	71.05	70.32	69.89	70.09	69.98	70.64	**70.31**

Table 4: This table reports the main results of our study. It shows the performance along different settings including the datasets, the training, decoding and span representations. We report the average across three seeds. **Bold** numbers indicate the best model/decoding for a fixed representation and underlined numbers indicate the best representation for a fixed model/decoding.

Dataset	P	R	F
Conll-2003	91.24	90.68	90.96
OntoNotes 5.0	87.80	88.92	88.36
TDM	69.77	73.65	71.66

Table 5: Performance for the baseline sequence labelling approach, a BERT-CRF tagger averaged over three seeds.

6.2 Evaluation metrics

Our evaluation is based on the exact match between predicted and gold entities. We report the micro-averaged precision (P), recall (R) and the F1-score (F) on the test set for models selected on the dev set.

6.3 Implementation Details

Backbones For span encoding, we used RoBERTa-base (Liu et al., 2019) for models trained on Conll-2003 and OntoNotes 5.0 because they come from general domains and we employed SciBERT (Beltagy et al., 2019) for models trained on TDM, which is a scientific NER data set.

Baseline model We compare the span-based approaches to a sequence labelling BERT-CRF (Beltagy et al., 2019), which we trained on our datasets.

Hyperparameters All models were trained using a single V100 GPU. We trained for up to 25 epochs using Adam (Kingma and Ba, 2017) as the optimizer with a learning rate of 1e-5. We opted for a batch size of 10 and used early stopping with a patience of 5 (on the F1-score) and keep the best model on the validation set for testing.

Libraries We implement our model with pytorch (Paszke et al., 2019). The pre-trained transformer models were loaded from the HuggingFace's Transformers (Wolf et al., 2020). We employed AllenNLP (Gardner et al., 2018) for data preprocessing and the seqeval library (Nakayama, 2018) for evaluating the baseline sequence labelling model. Our Semi-CRF implementation is based on pytorch-struct (Rush, 2020).

7 Results

7.1 Span Representation

In the following, we analyze the performance of the span representations on both the local model and the Semi-CRF model, as shown in the table 4.

Local models On local models, we find that *Convolution*, *Endpoints* and *Maxpool* all got competitive results while *FirstToken* representation obtains

Dataset	Model	#Examples						
		100	250	500	1000	2500	5000	All
Conll-2003	CRF	68.92	**77.49**	**82.05**	**85.38**	**88.50**	**89.40**	**90.96**
	Local	63.09	70.95	77.21	82.46	85.51	87.62	90.77
	+ Greedy	66.44	73.02	78.70	83.39	86.2	88.05	90.79
	+ MWIS	66.54	73.1	78.70	83.47	86.23	88.01	90.80
	+ EXT	65.54	72.53	78.49	83.35	86.10	88.00	90.82
	Semi-CRF	**69.21**	73.91	79.26	82.6	86.03	87.26	89.43
OntoNotes 5.0	CRF	61.0	69.59	74.17	77.18	78.86	81.08	88.36
	Local	60.23	68.13	73.33	76.69	81.32	82.49	88.80
	+ Greedy	62.60	70.20	74.95	77.46	**82.13**	83.09	88.96
	+ MWIS	**63.03**	**70.28**	**74.97**	**77.52**	82.08	**83.10**	88.94
	+ EXT	61.95	69.88	74.69	77.37	82.06	83.07	**88.98**
	Semi-CRF	63.02	69.46	72.79	77.03	80.33	81.97	87.81
TDM	CRF	**63.39**	**68.39**	**69.76**				**71.66**
	Local	54.87	63.1	67.08				68.78
	+ Greedy	55.94	65.28	67.64				71.26
	+ MWIS	57.04	65.22	67.60				71.48
	+ EXT	55.06	64.38	67.43				70.57
	Semi-CRF	60.49	65.06	66.52				71.05

Table 6: Few-shot performance. We report the average F1-score across three different seeds in all datasets and different training set sizes.

a result one notch below the others. On both the conll and TDM datasets, *Convolution* performed the best, yet the endpoints performed only slightly worse. However, on OntoNotes, the *Maxpool* representation outperforms all other approaches, while the *Endpoints* and *Convolution* got very similar performance. Out of all the datasets, *FirstToken* had the lowest score.

Global models On Semi-CRF models, the *Endpoints* representation consistently achieves the best results across datasets. We also notice that the *FirstToken* representation has better result than *Maxpool* and *Convolution* on two datasets, Conll-2003 and TDM in this setting.

The *Endpoints* representation is the most reliable overall, since it achieves robust performance regardless of the context in which it is used. However, for optimal performance and given a sufficient amount of compute resources, the span representation should be best tuned on a held-out set.

7.2 Comparison of Decoding for Local Models

Table 4 shows the performance results of the different decoding algorithms under different settings. For the local models, we can see that the application of decoding always improves the performance of the F1 score, by increasing the precision and by decreasing the recall score. However, there is no significant difference between the greedy decoding and the global decoding since the models are already well trained and thus, the overlap filtering does not make much difference in terms of quantitative results. We will provide more insight on decoding in the subsections 7.3 and 7.5.

7.3 Few-Shot Performance

We conducted a study to compare the performance of each model in a few-shot scenario. The evaluation was performed on the test set of each dataset using from 100 to the full training dataset. For this study, we used the Endpoints representation for spans because it is widely used and has shown good performance across different training and decoding schemes. The results of our few-shot evaluation are presented in Table 6.

Semi-CRF is better than the local spans-based approach when overlap filtering is not performed but the local approach performs better than Semi-CRF when the number of data become larger. Furthermore, while the difference between Greedy decoding and MWIS decoding is narrow in the high data regime, we can see that MWIS outperforms Greedy decoding in the low and very low data regime. Furthermore, we notice that the in-

	Conll			OntoNotes			TDM		
	P	R	F	P	R	F	P	R	F
Local	91.07	90.48	90.77	88.06	89.55	88.80	67.75	69.88	68.78
+ decoding	0.47	-0.37	+0.05	+0.89	-0.54	+0.18	+7.5	-1.76	+2.7
Neg. Sample	90.69	90.54	90.61	86.81	90.23	88.49	66.83	73.66	**70.01**
+ decoding	+0.84	-0.43	+0.21	+1.58	-0.98	+0.33	+7.7	-2.12	+2.97
Down-weighting	90.88	90.10	90.49	87.70	90.24	**88.95**	57.79	78.63	66.52
+ decoding	+0.48	-0.27	+0.1	+1.24	-0.72	+0.28	+13.77	-3.92	+6.57
Thresholding	90.80	90.96	**90.88**	87.49	88.81	88.14	63.99	74.56	68.85
+ decoding	+0.90	-0.41	+0.25	+1.00	-0.61	+0.21	+6.78	-2.70	+2.32

Table 7: Result for the local model when changing the training/loss. The best results before decoding are in **bold** and the best results after decoding are underlined. For this experiment, we use MWIS as decoding. We report the average over three seeds.

crease in performance by decoding is higher when a local model is training on a few datasets while the difference becomes less significant when the number of training data is large.

We find that the baseline sequence labelling, BERT-CRF approach is indeed competitive. It most of the time obtains a better performance on Conll-2003 and TDM datasets across any dataset sizes. However, the span-based approach is better on the OntoNotes 5.0 dataset. This can be explained by the fact that OntoNotes 5.0 contains 18 entity types and, therefore, the labelling approach would require 37 labels since it uses a BIO scheme, which makes the task much more difficult.

7.4 Analysis of Local Modeling

We previously found that decoding had little effect on our local model performance, especially for high resource datasets. We believe this is due to the fact that we were training with all negative samples (non-entity spans). As a result, the model was overconfident regarding non-entity spans (and not confident enough to predict entity spans) due to this unbalanced training. To resolve this issue, we propose three alternative training procedures to make the classifier leave more room for the decoder.

Negative sampling This approach randomly drops a percentage of the non-entity spans during training, but keeps all positive samples (entity spans). By training with fewer non-entity spans, we expect the model to be less confident and thus predict more entities. This negative sampling has been previously used by Li et al. (2021) to avoid training NER models with unlabeled (or missing) entities.

Down-weighing This method is similar to negative sampling, but instead of randomly eliminating negative samples, this approach retains all negative samples and down-weights their loss contribution while keeping loss for entity spans intact.

Thresholding This approach separates the span classifier into two models: a filtering model to classify whether a span is an entity or not, and a second an entity classification model to classify the entity type. During training, both models are trained end-to-end by multi-task learning with equally weighted losses. For prediction, span filtering is first performed and then the result is passed to the entity classification layer. By default, a span is passed into the entity classification layer if its probability of being an entity is greater than 0.5; however, we here adjust this threshold on the dev set and select the one with best F1 score.

The result from this analysis is show in the table 7. The results of this analysis show that, overall, the use of regularization techniques leads to a significant improvement in decoding accuracy for most datasets. As the most striking example, we can see that on the TDM dataset, the down-weighting approach which initially had a precision score of 57.79 was able to increase this score by 13.77 thanks to decoding improvements. Furthermore, it appears that the best approach according to these empirical results is the downw-eighting approach. Under this method, the decoder was most "successful" on both OntoNotes and TDM datasets, meaning it brought the largest improvements relatively to the performance of the local classifier before decoding.

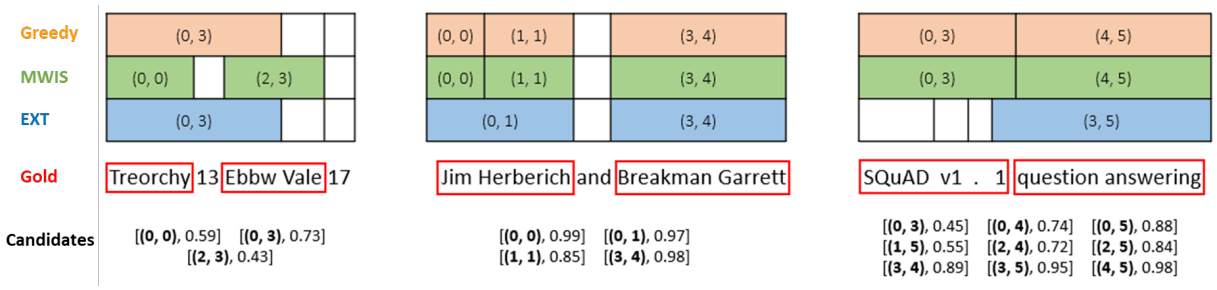

Figure 1: Shows how overlapping conflicts are handled by the different decoding algorithm on local span-based NER models. We only include overlaps involving at least three entities, because otherwise all decoding produce the same result.

7.5 Qualitative Comparison of Decoding

We performed a qualitative analysis to compare the three decoding approaches for local models. This study is presented in Figure 1, which shows the input text (truncated), the raw prediction with overlap, and the results after applying greedy decoding and the global decoding (MWIS and EXT). We only include overlaps involving more than two spans, because when two spans overlap, all algorithms take the span with the highest score.

We can see that the greedy approach always retrieves the most probable entity since it iteratively selects the best spans that do not overlap with previously selected spans. However, this algorithm tends to suffer from a myopic bias. Second, the MWIS approach, which maximizes the sum of span scores, tends to select as many spans as possible, which means that it favours shorter spans over longer ones. Also, MWIS decoding has a slightly higher recall score most of the time than other decoding algorithms. Finally, EXT decoding, which selects the set of spans that maximizes the average score, tends to select the smallest number of spans, but the selected spans generally have a high score. In general, this decoding tends to favour precision over recall score.

8 Related Works

Different approaches for NER NER is an important task in Natural Language Processing and is used in many downstream information extraction applications. Usually, NER tasks are designed as sequence labelling (Chiu and Nichols, 2016; Huang et al., 2015; Ma and Hovy, 2016; Lample et al., 2016; Strubell et al., 2017; Rei, 2017; Akbik et al., 2018) where the goal is to predict BIO tags. Recently, different approaches have been proposed to perform NER tasks that go beyond traditional sequence labelling. One approach that has been widely adopted is the span-based approach (Liu et al., 2016; Luan et al., 2018, 2019; Fu et al., 2021; Li et al., 2021; Zaratiana et al., 2022; Corro, 2022) where the prediction is done in the span level instead of entity level. Li et al. (2020) has also approached NER as a question answering task in which named entities are extracted by retrieving answer spans. In addition, recent work such as (Cui et al., 2021) considers NER as template filling by fine-tuning a BART (Lewis et al., 2019) encoder-decoder model.

Decoding For the spans-based approach, Semi-Markov has been used previously (Sarawagi and Cohen, 2005; Liu et al., 2016; Kong et al., 2016; Sato et al., 2017), however, their use with a BERT-type model has been little explored, something we did in this paper. The work of Fu et al. (2021) and Li et al. (2021) employed a heuristic decoding to avoid overlap for span-based NER. Their algorithm iteratively chooses the maximum probability entity span that does not overlap with a previously chosen entity span. In this paper, we have proposed an exact version of this algorithm.

9 Conclusion

We investigated different span representations for NER and found that the endpoint representation is the most robust. Moreover, we have proposed a new formulation of NER using overlapping graphs for which an exact and efficient decoding algorithm exists. We used the formulation to eliminate span overlap on locally trained models. Finally, we conducted few-shot performance analysis for different modelling approaches and found that classical sequence labeling models provide strong results for datasets with few entity types, while span-based approaches are better for larger type sets.

Acknowledgments

This work is partially supported by a public grant overseen by the French National Research Agency (ANR) as part of the program Investissements d'Avenir (ANR-10-LABX-0083). This work was granted access to the HPC/AI resources of [CINES/IDRIS/TGCC] under the allocation 20XX-AD011013096 made by GENCI.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: A pretrained language model for scientific text.

Xavier Carreras, Lluís Màrquez, and Lluís Padró. 2002. Named entity extraction using AdaBoost. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*.

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.

Caio Corro. 2022. A dynamic programming algorithm for span-based nested named-entity recognition in $o(n^2)$. *ArXiv*, $abs/2210.04738$.

Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. 2021. Template-based named entity recognition using bart.

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.

Markus Eberts and Adrian Ulges. 2020. Span-based joint entity and relation extraction with transformer pre-training. *ArXiv*, abs/1909.07755.

Jinlan Fu, Xuanjing Huang, and Pengfei Liu. 2021. SpanNER: Named entity re-/recognition as span prediction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7183–7195, Online. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform.

Yufang Hou, Charles Jochim, Martin Gleize, Francesca Bonin, and Debasis Ganguly. 2021. TDMSci: A specialized corpus for scientific literature entity tagging of tasks datasets and metrics. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 707–714, Online. Association for Computational Linguistics.

Ju Yuan Hsiao, Chuan Yi Tang, and Ruay Shiung Chang. 1992. An efficient algorithm for finding a maximum weight 2-independent set on interval graphs. *Information Processing Letters*, 43(5):229–235.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging.

David S Johnson, Mihalis Yannakakis, and Christos H Papadimitriou. 1988. On generating all maximal independent sets. *Information Processing Letters*, 27(3):119–123.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Segmental recurrent neural networks. *CoRR*, abs/1511.06018.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.

Yanfei Lei, Chunming Hu, Guanghui Ma, and Richong Zhang. 2021. Keyphrase extraction with incomplete annotated training data. In *Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021)*, pages 26–34, Online. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020. A unified mrc framework for named entity recognition.

Yangming Li, lemao liu, and Shuming Shi. 2021. Empirical analysis of unlabeled entity problem in named entity recognition. In *International Conference on Learning Representations*.

Yaoyong Li, Kalina Bontcheva, and Hamish Cunningham. 2004. Svm based learning system for information extraction. In *Deterministic and Statistical Methods in Machine Learning*.

Yijia Liu, Wanxiang Che, Jiang Guo, Bing Qin, and Ting Liu. 2016. Exploring segment representations for neural segmentation models. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 2880–2886. AAAI Press.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232, Brussels, Belgium. Association for Computational Linguistics.

Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 2019. A general framework for information extraction using dynamic span graphs.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf.

Hiroki Nakayama. 2018. seqeval: A python framework for sequence labeling evaluation. Software available from https://github.com/chakki-works/seqeval.

Madhumangal Pal and GP Bhattacharjee. 1996. A sequential algorithm for finding a maximum weight k-independent set on interval graphs. *International Journal of Computer Mathematics*, 60(3-4):205–214.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach 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.

Venkatesh Raman, Saket Saurabh, and Somnath Sikdar. 2007. Efficient exact algorithms through enumerating maximal independent sets and other techniques. *Theory of Computing Systems*, 41(3):563–587.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. *arXiv preprint arXiv:1704.07156*.

Alex Rogozhnikov. 2022. Einops: Clear and reliable tensor manipulations with einstein-like notation. In *International Conference on Learning Representations*.

Alexander M. Rush. 2020. Torch-struct: Deep structured prediction library.

Sunita Sarawagi and William W Cohen. 2005. Semi-markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems*, volume 17. MIT Press.

Motoki Sato, Hiroyuki Shindo, Ikuya Yamada, and Yuji Matsumoto. 2017. Segment-level neural conditional random fields for named entity recognition. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 97–102, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and accurate sequence labeling with iterated dilated convolutions.

Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. 2021. Synthesizer: Rethinking self-attention in transformer models. *ArXiv*, abs/2005.00743.

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.

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 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*, 23.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *ACL*.

Urchade Zaratiana, Nadi Tomeh, Pierre Holat, and Thierry Charnois. 2022. GNNer: Reducing overlapping in span-based NER using graph neural networks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 97–103, Dublin, Ireland. Association for Computational Linguistics.

Zexuan Zhong and Danqi Chen. 2021. A frustratingly easy approach for entity and relation extraction.