A Structured Span Selector

Tianyu Liu **Yuchen Eleanor Jiang** **Ryan Cotterell** **Mrinmaya Sachan**
ETH Zürich
{tianyu.liu, yucjiang, ryan.cotterell, mrinmaya.sachan}@inf.ethz.ch

Abstract

Several natural language processing tasks such as coreference resolution and semantic role labeling require selecting text spans and making decisions upon them. A typical approach for these tasks is to naively score all spans and greedily select spans for downstream processing. This approach ignores any useful linguistic biases such as spans that need to be considered for these problems are typically syntactic constituents. In this paper, we propose a novel linguistically-motivated "structured" span-selection model which learns to make use of the partial span-level annotation provided for such problems. Compared to previous approaches, our approach gets rid of the heuristic greedy span selection scheme, allowing us to model the downstream task on an optimal set of spans. We evaluate our model on two popular span prediction tasks: coreference resolution and semantic role labeling; and show improvements on both tasks in terms of performance as well as efficiency.

1 Introduction

Span selection, i.e., the problem of selecting a continuous segment (a **span**) of the input text, is a common design pattern[1] in many span prediction problems in NLP. Common span prediction problems include coreference resolution (Stede, 2011) where the selected spans are mentions, semantic role labeling (Palmer et al., 2010) where the selected spans are arguments, question answering (Jurafsky and Martin, 2000) where the selected spans are answers, and named entity recognition (Smith, 2011) where the selected spans are entities. The span selection step is usually followed by a task specific model. For instance, in coreference resolution, the selected spans (mentions) are clustered according to the entity they refer to; whereas, in semantic role labeling the spans (arguments) are classified into a set of roles. An example of two span selection tasks is shown in Fig. 1.

Coref: [The President]$_1$ has said $\big[$[he]$_1$ and [his]$_1$ wife, now a [New York]$_3$ senator$\big]_2$ will spend weekends at [their]$_2$ house in Chappaqua.

SRL: [The most important thing about Disney]$_{1\text{-ARG1}}$ $\boxed{\text{is}_1}$ [that [it]$_{2\text{-ARG1}}$ $\boxed{\text{is}_2}$ [a global brand]$_{2\text{-ARG2}}$]$_{1\text{-ARG2}}$.

Figure 1: Examples of two span selection tasks: *Coreference* and *SRL*. In Coref, $[s]_i$ denotes a span of text s referring to the ith entity. In SRL, $\boxed{\text{i}}$ denotes a set of predicates and $[s]_{i-*}$ denotes a set of arguments for the ith predicate.

In most of the tasks mentioned above, span selection is the first step in a pipeline.[2] After a set of candidate spans is determined, a classifier (often a neural network) is typically used to score the candidate spans. As the number of spans to consider in the input text can be quadratic in the length of input, these spans are greedily selected as potential antecedents, roles or answers. While greedy span selection has become the de-facto approach in span prediction problems, it has several issues. First, such approaches typically ignore the inherent structure in the problem. For example, spans of interest in problems such as coreference and SRL are typically syntactic constituents, an assumption supported by quantitative results[3]. Lack of linguistic constraint on the spans of interest leads to a waste of computational resource, as all $\mathcal{O}(n^2)$ possible spans are enumerated by the model.

Thus, in this paper, we propose a **structured span selector** that solves the aforementioned problems. Our span selector is modeled as a chart parser which learns to score partial, possibly nested, span

[1] A software engineering metaphor, which refers to a reusable solution to a commonly occurring problem.

[2] As opposed to joint models where we find optimal spans and make downstream decisions simultaneously (Lee et al., 2017; He et al., 2018).

[3] For the OntoNotes dataset, He et al. (2018) reports that **98.7%** of the arguments in SRL are constituents. For coreference, we find that **99.1%** of the mentions are constituents.

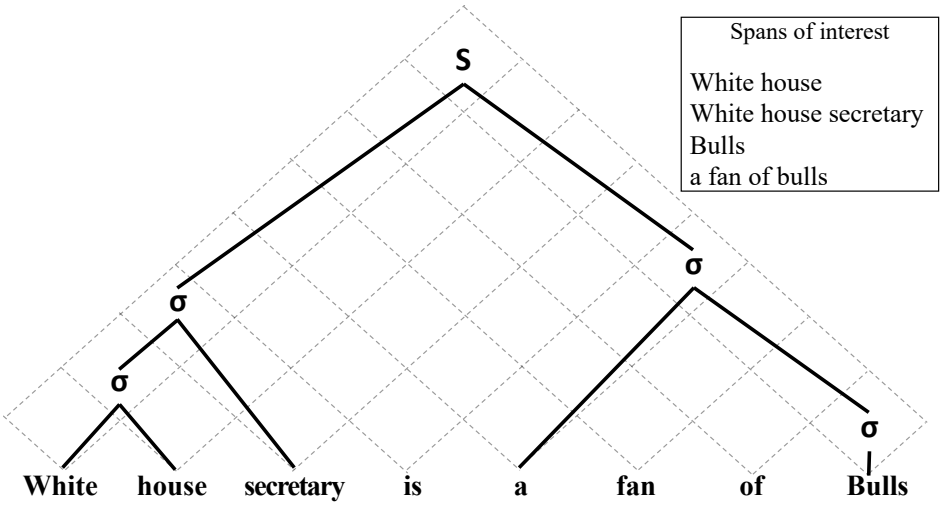

Figure 2: An example parse in our WCFG for coreference resolution and the set of spans it corresponds to. Production rules that do not have the "span of interest" non-terminal (σ) on the right-hand side are skipped as they do not affect the parsing result (see §3.1). Traditional greedy approach considers $\mathcal{O}(n^2)$ spans denoted by the grid cells unless pruning strategy is applied. While the number of nonterminals in a CNF parse is $\mathcal{O}(n)$.

annotations given in the training corpus. Instead of optimizing the full parse, we directly learn to estimate the anchored probability of constituents using the inside scores in the parse. The anchored probability can be easily computed by backpropagating through the inside score (Eisner, 2016). Fig. 2 illustrates an example parse of our WCFG and the difference between the traditional greedy approach and our approach.

We apply our span selector to one coreference (Joshi et al., 2020) and one SRL model (He et al., 2018). In both cases, we optimize the log-likelihood of the joint distribution defined by the span selector and the conditional for the downstream task as defined in Lee et al. (2017) and He et al. (2018). In contrast to previous approaches which use a set of heuristics[4] to prune the set of spans to be considered, our span selector directly generates the optimal set of spans. The number of resulting spans are significantly lower than the number of spans considered in previous works, resulting in reduction in memory required to train these models. The selected spans are also more accurate. The proposed approach leads to significant gains in both downstream tasks considered in this work: coreference and SRL. We find that our approach improves the performance of the end-to-end coreference model on the OntoNotes English dataset (by 0.4 CoNLL F1 points) and the LitBank dataset (by 0.7 CoNLL F1 points). On SRL, our model also achieves consistent gains over previous work.

[4]Lee et al. (2017) consider spans such that the maximum span width is 10, the number of spans per word is 0.4, etc. Then, the spans are greedily pruned.

2 Background and Related Work

2.1 Span Selection as a Design Pattern

Many NLP tasks involve reasoning with textual spans. Such tasks include coreference resolution, semantic role labelling, named entity recognition and question answering. Models for these span prediction tasks often follow a common design pattern. The architectures for these models often decompose into two components: (i) a **span selection** component where the model first selects a set of spans of interest, and (ii) a **span prediction** component where a prediction (e.g. entity or role assignment) is made for the chosen set of spans.

As shown in previous works (Zhang et al., 2018; Wu et al., 2020), the quality of the span selector can have a large impact on the overall model performance. The span selector typically selects spans by selecting the start token and the end token in the span. Thus, there are inherently $\binom{N}{2}$ textual spans (the complexity is $\mathcal{O}(n^2)$) within a document with N tokens to choose from. Previous span selection models (Lee et al., 2017; He et al., 2018) enumerate all possible spans in a brute force manner and feed greedily selected top-k span candidates for the downstream prediction step.

However, several span selection tasks require spans that are syntactic constituents, which is a useful but often neglected inductive bias in these

works. For instance, in coreference resolution, span selection models are supposed to extract noun phrases, pronouns, and sometimes, verbs. While in semantic role labeling, span selection models are required to extract syntactic constituents that could form semantic arguments of a predicate, such as noun phrases, prepositional phrases, and adverbs.

Our work uses a context-free parsing formulation to enumerate spans. In context-free grammars, the number of valid syntactic constituents in a sequence of length n is bounded by $\mathcal{O}(n)$, as the constituents can be viewed as the internal nodes of a binary parse tree in Chomsky normal form. Compared with brute-force enumeration and greedy pruning, this inductive bias serves us as a natural pruning strategy to reduce the complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$ and provides us a more natural and linguistically informed way to model span-based tasks in NLP, through which we can employ parsing techniques and retrieve the optimal span selection for downstream tasks.

To further motivate our work, we provide a background on two popular span prediction tasks considered in our work: coreference resolution and semantic role labelling (SRL). We also describe some previous works on these tasks and contrast them to our work. Then, we also describe how our model can work with partial span-level annotations provided by datasets for these tasks.

2.2 Coreference Resolution

Most coreference resolution models involve two stages: mention detection and mention clustering. Traditional pipeline systems relied on parse trees and hand-engineered rules to extract mentions (Raghunathan et al., 2010). However, this was reversed by Lee et al. (2017) who showed that we can directly detect mentions as well as assign antecedents for them in an end-to-end manner.

Alongside our work, some other works (Zhang et al., 2018; Swayamdipta et al., 2018; Miculicich and Henderson, 2019; Yu et al., 2019) have also explored better mention proposers for coreference. Zhang et al. (2018) use multi-task loss to directly optimize the mention detector. Swayamdipta et al. (2018) also leverage syntactic span classification as an auxiliary task to assist coreference. Miculicich and Henderson (2019) and Yu et al. (2019) both improve the mention detector with better neural network structures. Yet they still need to manually set a threshold to control the number of selected candidate mentions and non of them could produce an optimal span selection for downstream task. Finkel and Manning (2009) situate NER in a parsing framework by explicitly incorporating named entity types into parse tree labels. In contrast, our work requires neither syntax annotations nor hyperparameter tuning for mention selection.

Zhang et al. (2018) proposed a multi-task style loss to provide direct supervision to mention detection. Yet the seemingly simplistic and heuristic mention detection scheme by greedily taking a fixed top portion of all candidate spans still remains unchanged in the aforementioned work.

Pretrained contextual word representations (Joshi et al., 2019, 2020; Lee et al., 2018) and higher-order inference techniques (Lee et al., 2018; Kantor and Globerson, 2019; Xu and Choi, 2020) have brought major improvements to the original model by Lee et al.. Aside from this line of work, Wu et al. (2020) formulate coreference resolution as a question answering task, and achieve state-of-the-art results on multiple benchmarks.

2.3 Semantic Role labeling

Semantic role labeling (SRL) extracts relations between predicates and their arguments. Two major lines of work in SRL are sequence tagging models (He et al., 2017; Marcheggiani et al., 2017) and span-based models (He et al., 2018; Ouchi et al., 2018; Li et al., 2019).

Sequence tagging models for SRL convert semantic role annotations to BIO sequences. The tagger generates a label sequence for one single predicate at a time. However, span-based models generate the set of all candidate arguments in one forward pass, and classify their semantic roles with regard to each predicate. As discussed in (He et al., 2018), span-based models empirically perform better than sequence tagging models as they incorporate span-level features. Span-based models also do better at long-range dependencies as well as agreement with syntactic boundaries.

Very recently, Zhang et al. (2021) explore a novel direction by reducing semantic role labeling to a dependency parsing task and find promising results.

2.4 Nested and Partial Span Annotations

Nested mentions and partially annotated mentions are the two major concerns of this work. Most datasets for span prediction problems contain partial annotations of mentions. For example, singletons are not annotated in Ontonotes (Pradhan et al.,

2012). In the coreference resolution example given in Fig. 1, the bracketed nested spans are annotated, while the underlined spans are valid mentions that are missed out, since they do not co-refer with any other mention in the same document.

The same issues also exist in SRL. In the SRL example in Fig. 1, there are two predicates (boxed words in the example) in one sentence. Their arguments are nested (i.e. ARG1 and ARG2 of the second predicate locate within ARG2 of the first predicate). Besides, many syntactic constituents that could form valid arguments are unannotated.

Our span selector can be trained with partial span annotations provided in the training data. See §4 for further details.

3 A Structured Span Selection Model

We first start by giving a formal definition of a span selection task. We define a **document** D as a sequence of sentences $\mathbf{w}_1, \ldots \mathbf{w}_{|D|}$. Each **sentence** $\mathbf{w}$ in the document is a sequence of words $[w_1, \ldots, w_{|\mathbf{w}|}]$. A **span** is a contiguous subsequence of words in a sentence. For instance, we denote the span from position i to position k as $w_i \cdots w_k$, which we abbreviate as σ_{ik}.

3.1 A Weighted Context-Free Grammar

In this section, we formalize the linguistic inductive bias that *a majority of spans considered in span selection tasks are constituents*. Under this assumption, the set of selected spans in a sentence should be compatible with a phrase structure tree[5], i.e., there should not be any spans crossing each other in the selected set of spans.

To model this tree structure, we propose to learn a weighted context-free grammar (WCFG) $\langle \Sigma, N, S, R \rangle$, where Σ is a finite set of terminal symbols, N is a finite set of nonterminal symbols, $S \in N$ is the start symbol, and R is the set of production rules. A learnable weight function $s_p : R \to \mathbb{R}_{\geq 0}$ assigns a non-negative weight to each rule in R. This function naturally extends to assign a weight to each CNF **parse** t by

$$s_p(t) = \prod_{r \in t} s_p(r) \tag{1}$$

where all internal nodes of t are enumerated. Most importantly, our WCFG defines a distribution over the set of all parses of a sentence $\mathcal{T}$ given by

$$p(t \mid \mathbf{w}) = \frac{s_p(t)}{Z} \tag{2}$$

where $Z = \sum_{t \in \mathcal{T}} s_p(t)$ is the sum of the scores of all parses.

Despite its formulation, we design our WCFG to be succinct — we only care about *two types of nonterminals* other than the start symbol S, that is, is a **span of interest** in the downstream task or **not**. Due to this, among all the production rules, only those with spans of interest on the right-hand side can affect the weight of a derivation[6]. In a Chomsky Normal Form derivation, a span of interest appears *once and only once* on the right-hand side. In this respect, a parse t can be uniquely mapped to *a set of spans of interest* by gathering all nonterminals of that type. We denote the resulting set after mapping as $M = \texttt{span}(t)$, and the set of all sets of spans of interest as $\mathcal{M}$.

Thus, the score of a parse can be re-expressed as a product of the scores of all spans of interest in it:

$$s_p(M) = \prod_{\sigma_{ik} \in M} s_p(\sigma_{ik}) \tag{3}$$

The problem of mention detection is hence converted from subset selection to finding an optimal parse tree that maximizes the score:

$$M^* = \operatorname*{argmax}_{M \in \mathcal{M}} s_p(M) = \texttt{span}\left(\operatorname*{argmax}_{t \in \mathcal{T}} s_p(t)\right) \tag{4}$$

where the classical algorithm for parsing, i.e., the CKY algorithm, can be employed to obtain an *exact* solution. The recursive function when $i < k$ used is:

$$\gamma([i,k]) = s_p(\sigma_{ik}) + \max_{i \leq j < k} \{\gamma([i,j]) + \gamma([j,k])\} \tag{5}$$

where $\gamma([i,k])$ is the score of the optimal derivation over span $[i,k]$. When $i = k$, $\gamma([i,k]) = s_p(\sigma_{ik})$.

We now briefly introduce the inside-outside algorithm to pave the way for the next section. For a span $[i,k]$, its inside value can be expressed as

$$\beta([i,k]) = \exp s_p(\sigma_{ik}) \cdot \left(\sum_{j=i}^{k-1} \beta([i,j]) \cdot \beta([j,k]) \right) \tag{6}$$

When $i = k$, $\beta([i,k]) = \exp s_p(\sigma_{ik})$. The inside value of the entire sentence $\beta([0,n])$ is exactly Z, the sum of scores of all parses.

[5] Fig. 1 shows two examples.

[6] See App. A for the details of our design choice.

3.2 Selecting Spans from a WCFG

We have been using the notation of **anchored nonterminal** σ_{ik}, when a parse contains a span of interest that ranges over $[i, k]$, without formally introducing it. Further, an **anchored probability**, i.e., how likely span $[i, k]$ would be of interest in any parse, can be computed.

Let $\mathcal{M}_{ik}$ be the set of parses that contains σ_{ik}, the anchored probability of σ_{ik} can be expressed as:

$$\begin{aligned} p(\sigma_{ik} \mid \mathbf{w}) &= \sum_{M \in \mathcal{M}_{ik}} p(M \mid \mathbf{w}) \\ &= \sum_{M \in \mathcal{M}_{ik}} \frac{\exp s_p(M)}{Z} \end{aligned} \tag{7}$$

As described by Eisner (2016), we can calculate $p(\sigma_{ik} \mid \mathbf{w})$ by backpropagating through the log-transformed inside value $\log Z = \log \beta([0, n])$:

$$p(\sigma_{ik} \mid \mathbf{w}) = \frac{\partial \log Z}{\partial s_p(\sigma_{ik})} \tag{8}$$

which can be efficiently computed with the automatic differentiation in modern deep learning tools such as PyTorch. For details of deduction of Eq. (8), we refer the readers to (Eisner, 2016, Sec. 4.3). We use $p(\sigma_{ik})$ as shorthand for $p(\sigma_{ik} \mid \mathbf{w})$ without causing ambiguities from now.

4 Adaptations to Downstream Tasks

In this section, we introduce how our structured span selector can be applied in end-to-end frameworks for coreference resolution and SRL.

4.1 Coreference Resolution

The goal of coreference resolution is to link every mention σ_{ij} to its **antecedent**, which is itself another mention in the same document. Note that in the formulation of coreference discussed in this task, a mention may have a dummy antecedent, denoted as ϵ, if it is not a mention or does not co-refer with any previous span.

One core technical contribution of this paper is a joint model for mention detection and downstream tasks. We write $\sigma_{ij} \rightsquigarrow \sigma_{k\ell}$ to denote that σ_{ij} co-refers with $\sigma_{k\ell}$; in other words, that $\sigma_{k\ell}$ is σ_{ij}'s antecedent. Our paper introduces the following framework for joint modeling mention detection and coreference resolution:

$$\underbrace{p(\sigma_{ij})}_{\text{pr. } \sigma_{ij} \text{ is a mention}} \times \underbrace{p\left(\sigma_{ij} \rightsquigarrow \sigma_{k\ell} \mid \sigma_{ij}\right)}_{\text{pr. } \sigma_{ij}\text{'s antecedent is } \sigma_{k\ell}}$$

From now, we use σ_{ij} as a shorthand for σ_{ij}, and $p\left(\sigma_{ij} \rightsquigarrow \sigma_{k\ell}\right)$ for $p\left(\sigma_{ij} \rightsquigarrow \sigma_{k\ell} \mid \sigma_{ij}\right)$. The joint distribution is then $p\left(\sigma_{ij} \rightsquigarrow \sigma_{k\ell}, \sigma_{ij}\right) = p\left(\sigma_{ij} \rightsquigarrow \sigma_{k\ell}\right) p(\sigma_{ij})$. In principle, modelers can choose $p(\sigma_{ij})$ and $p\left(\sigma_{ij} \rightsquigarrow \sigma_{k\ell}\right)$ according to their taste. In this work, we implement $p(\sigma_{ij})$ as the anchored probability in a WCFG, as described in §3.1 and $p\left(\sigma_{ij} \rightsquigarrow \sigma_{k\ell}\right)$ as Lee et al. (2017)'s popular coreference model, described in §4.1.

Lee et al. (2017) as Mention Linker. We choose $p\left(\sigma_{ij} \rightsquigarrow \sigma_{k\ell}\right)$ to be the one from (Lee et al., 2017). In principle, we can use any mention-ranking model that computes a distribution over $\mathcal{Y}([i, j])$, the set of preceding spans of $[i, j]$. But we choose (Lee et al., 2017) because it is very well known and effective. Using $\mathcal{Y}_{ij}$ as a shorthand for $\mathcal{Y}([i, j])$:

$$p\left(\sigma_{ij} \rightsquigarrow \sigma_{k\ell}\right) = \frac{\exp\left(s(\sigma_{ij}, \sigma_{k\ell})\right)}{\sum_{m \in \mathcal{Y}_{ij}} \exp\left(s\left(\sigma_{ij}, m\right)\right)}$$

$$s(\sigma_{ij}, \sigma_{k\ell}) = s_m(\sigma_{ij}) + s_m(\sigma_{k\ell}) + s_a(\sigma_{ij}, \sigma_{k\ell})$$

$s_m(\sigma_{ij})$ is a score for span $[i, j]$ being a mention and $s_a\left(\sigma_{ij}, \sigma_{k\ell}\right)$ for $[k, \ell]$ to be a coreferent antecedent of $[i, j]$. Both s_a and s_m are computed by neural networks that take span representations as inputs. The score $s(\sigma_{ij}, \epsilon)$ is fixed to 0.

Training, Decoding and Other Concerns. As for mention detection, Lee et al.'s model adopts a naïve greedy algorithm by taking the top $\lambda|D|$ spans with the highest mention scores s_m, where λ is a hyperparameter that has to be manually defined for different datasets. However, finding a proper ratio λ can be very tricky. Since in many coreference datasets such as OntoNotes, only the mentions that are referred to more than once are annotated (as discussed in §3.1), learning a mention detector from such data requires the ability to handle partially annotated examples.

The case that $[i, j]$ is assigned to an antecedent assignment ϵ (i.e. $y_{ij} = \epsilon$) happens when (a) $[i, j]$ is a singleton mention, or (b) $[i, j]$ is the first appearance in a mention cluster, or (c) $[i, j]$ is not a valid mention. To handle the partial annotation and retrieve valid mentions that are not annotated in the dataset, we split the probability into two separate terms corresponding to the above scenarios:

$$p(\sigma_{ij} \rightsquigarrow \epsilon, \sigma_{ij}) = p(\sigma_{ij}) p(\sigma_{ij} \rightsquigarrow \epsilon) + (1 - p(\sigma_{ij})) \tag{9}$$

We expect singleton mentions to have a larger first term and smaller second term, and the opposite

for non-mentions. The final objective function is then the negative log-likelihood of the joint distribution defined at the beginning of §4.1:

$$L = -\sum_{\mathbf{w}\in D}\sum_{[i,j]\in\mathbf{w}}\left(\log\sum_{m\in\mathcal{Y}_{ij}\cap\mathcal{G}_{ij}} p(\sigma_{ij}\rightsquigarrow m,\sigma_{ij})\right) \tag{10}$$

where $\mathcal{G}_{ij}$ is the ground truth mention cluster that span $[i,j]$ belongs to.

For mention detection, the time complexity of training and inference for one sentence is $\mathcal{O}(n)$ with parallel computation. Sentence-level parallelism can also be applied to all the sentences. Thus, the overall time complexity is $\mathcal{O}(n_{\text{max}})$, the length of the longest sentence in the document.

4.2 Semantic Role Labeling

The goal of semantic role labeling is to classify the **semantic role** of every argument σ_{ij} with respect to a given predicate. Following the notation style of §4.1, we use $\sigma_{ij} \overset{l}{\rightsquigarrow} v$ to denote that σ_{ij} plays the semantic role l in the frame of predicate v. The joint probability can be written as $p\left(\sigma_{ij}\overset{\ell}{\rightsquigarrow} v,\sigma_{ij}\right) = p\left(\sigma_{ij}\overset{\ell}{\rightsquigarrow} v\right)p(\sigma_{ij})$. Our joint framework for argument detection and semantic role labeling is then:

$$\underbrace{p(\sigma_{ij})}_{\text{pr. } \sigma_{ij} \text{ is an argument}} \times \underbrace{p\left(\sigma_{ij}\overset{\ell}{\rightsquigarrow} v \mid \sigma_{ij}\right)}_{\text{pr. } \sigma_{ij}\text{'s role is } \ell \text{ w.r.t. predicate } v}$$

He et al. (2018) as Role Classifier. As this work focuses on span selection, we choose the role classifier to be the popular and effective one from (He et al., 2018), and use gold predicates v during training and evaluation.

The semantic role label l takes value from a discrete label space $\mathcal{L}$. The classifier then models the following probability distribution:

$$p\left(\sigma_{ij}\overset{\ell}{\rightsquigarrow} v\right) = \frac{\exp\left(s(\sigma_{ij},v,\ell)\right)}{\sum_{\ell\in\mathcal{L}}\exp\left(s(\sigma_{ij},v,\ell)\right)}$$
$$s(\sigma_{ij},v,\ell) = s_m(\sigma_{ij}) + s_r(\sigma_{ij},v,\ell)$$

Similar to the coreference model of Lee et al. (2017), $s_m(\sigma_{ij})$ is the score for span $[i,j]$ to be an argument and $s_r\left(\sigma_{ij},v,\ell\right)$ for $[i,j]$ to play the role l for predicate v. Note that the score for ϵ label (i.e. no relation) $s(\sigma_{ij},v,\epsilon)$ is set to constant 0, similar to the null antecedent case in coreference.

Training Objective. He et al.'s model also suffers from the challenge of tuning the hyperparameter λ and the issue of partial annotation for the span selection model. To deal with these, we use a similar strategy as in §4.1. Again, the probability of null relation is split into two terms:

$$\begin{aligned} &p(\sigma_{ij}\overset{\epsilon}{\rightsquigarrow} v,\sigma_{ij}) \\ &\quad = p(\sigma_{ij})p(\sigma_{ij}\overset{\epsilon}{\rightsquigarrow} v) + (1-p(\sigma_{ij})) \end{aligned} \tag{11}$$

Thus, the training objective for SRL with our structured span selection model is then:

$$L = -\sum_{[i,j]\in S}\sum_{v}\log p(\sigma_{ij}\overset{l}{\rightsquigarrow} v,\sigma_{ij}) \tag{12}$$

where l is the correct semantic label of σ_{ij} with regard to predicate v.

5 Experiments

5.1 The Greedy Baseline

Previous work has mostly considered a greedy procedure for span selection as opposed to Eq. (4). The approach produces a score $s_g(\sigma_{ik})$ *independently* for each span σ_{ik}. As the number of spans σ_{ik} is potentially very large, the set of spans are greedily pruned to a set of size K. For instance, in SRL, where spans are selected for each sentence:

$$M^*_{\text{topk}} = \text{atop}_K\left(\left\{s_g(\sigma_{ik}) \mid 1\le i<k\le|\mathbf{w}|\right\}\right) \tag{13}$$

However, in coreference resolution, a set of spans are selected for the entire document:

$$M^*_{\text{topk}} = \text{atop}_K\left(\bigcup_{\mathbf{w}\in D}\left\{s_g(\sigma_{ik}) \mid 1\le i<k\le|\mathbf{w}|\right\}\right) \tag{14}$$

where atop_K is shorthand for argtop_K.

We will see in our experiments that tuning K can be quite challenging. Moreover, as the greedy approach scores each span independently, it cannot judiciously make use of the provided span annotation.

5.2 Datasets

Coreference: We employ the CoNLL-2012 English shared task dataset (OntoNotes) (Pradhan et al., 2012) and LitBank (Bamman et al., 2020) in our experiments. For OntoNotes, we apply the speaker encoding in (Wu et al., 2020), that is using special tokens (`<speaker>`, `</speaker>`) to denote the speaker's name, as opposed to the original binary features used by Lee et al. (2017). This

simple change brings a consistent boost to the performance by 0.2 CoNLL F1. A major difference between these two datasets is that LitBank has singleton mention annotations while OntoNotes does not. For LitBank, we use the standard 10-fold cross-validation setup, as is the standard practice.
SRL: We use the CoNLL-2012 SRL dataset. Gold predicates are provided to the model.

5.3 Coreference Resolution

We report the average precision, recall, and F1 scores of the standard MUC, B^3, and $CEAF_{\phi_4}$ and the average CoNLL F1 score on OntoNotes test set in Tab. 1. The average CoNLL F1 scores on LitBank are shown in Tab. 2. For OntoNotes, we run the experiments with 5 random initializations and the improvements reported are significant under the two-tailed *t*-test.

We compare our models with several representative previous work. In order to focus on comparing the impact of mention detection, we apply no higher-order inference techniques in our models and report the non-higher order result from (Xu and Choi, 2020). Joshi et al. (S) is the major baseline that uses SpanBERT (Joshi et al., 2020) and is trained with the speaker encoding discussed in §5.2. This encoding gives 0.2 F1 score improvement over the result reported in (Xu and Choi, 2020).

Our model with the structured mention detector achieves 80.5 F1 score, a 0.4 F1 improvement over the baseline. While on LitBank, our model achieves 76.3 F1, which is a 0.7 F1 improvement over Joshi et al.. It can also be observed that this gain mainly comes from improved recall, which is because we have a superior mention detector that can retrieve mentions with better accuracy. We further analyze this result in the following section.

5.3.1 Analysis of Mention Detector

Next, we examine the performance of our proposed mention detection scheme. As shown in Fig. 3, compared with Joshi et al. (S), our model predicts a far more accurate set of mentions with a higher recall. In contrast to Joshi et al. (S) who select $0.4|D|$ mention spans, our method only selects $0.26|D|$ mention spans with a slightly higher mention recall. The smaller span set would make the overall coreference model more efficient as well.

5.3.2 Analysis of Structured Modeling

To see how our structured modeling benefits the overall coreference model, we further compare our

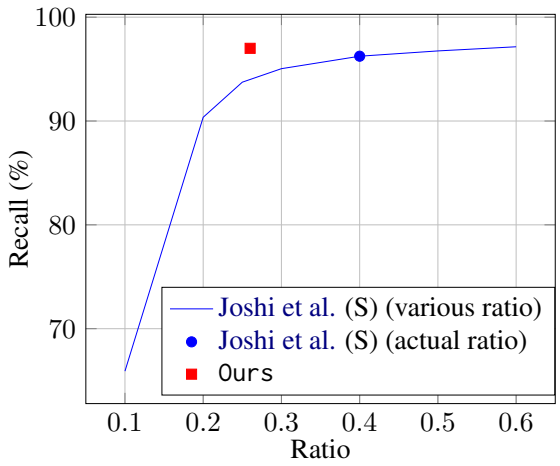

Figure 3: Recall of gold mentions as we vary the ratio of spans kept. Ratio refers to the number of predicted mentions divided by $|D|$. Our mention detector significantly outperforms Joshi et al. (S)'s with a ratio of 0.26 and a recall of 97.0%. Yet Joshi et al. (S) only achieves 96.2% recall with a ratio of 0.40.

approach with a baseline `Sigmoid` which replaces the $p(\sigma_{ij})$ in Eq. (8) with a simple non-structured estimator:

$$p_\sigma(\sigma_{ij}) = \sigma(s_p(\sigma_{ij})) \tag{15}$$

where σ is the sigmoid function $\sigma(x) = \frac{1}{1+\exp(-x)}$. The loss function used for `Sigmoid` is identical with our structured model as given in Eq. (10). Through this comparison, we aim to show the effectiveness of structured modeling.

We also build a multi-task learning baseline `MTL` similar to (Swayamdipta et al., 2018). The model performs an auxiliary task which classifies spans into noun phrase, other syntactic constituent, or non-constituent. The coefficient of the multi-task loss is set to 0.1 as in (Swayamdipta et al., 2018).

This comparison is shown in Tab. 3. We find that replacing $P(i)$ with unstructured $P_\sigma(i)$ degrades the performance by 0.3 F1 score. Thus, we conclude that the structured probability function is more expressive than $P_\sigma(i)$ as it models the global annotation for each sentence. In contrast, the mention detector of Joshi et al. (S) and `Sigmoid` models each span independently.

5.3.3 Qualitative Analysis

To shed some light on where the gains of our proposed model come from, we also exhibit some qualitative results in this section.

Nested Mentions. We first investigate the capability of our structured model in handling nested

	MUC			B^3			$CEAF_{\phi_4}$			
	P	R	F	P	R	F	P	R	F	Avg. F1
Lee et al. (2017)	78.4	73.4	75.8	68.6	61.8	65.0	62.7	59.0	60.8	67.2
Lee et al. (2018)	81.4	79.5	80.4	72.2	69.5	70.8	68.2	67.1	67.6	73.0
Fei et al.	85.4	77.9	81.4	77.9	66.4	71.7	70.6	66.3	68.4	73.8
Kantor and Globerson	82.6	84.1	83.4	73.3	76.2	74.7	72.4	71.1	71.8	76.6
Joshi et al. (2019)	84.7	82.4	83.5	76.5	74.0	75.3	74.1	69.8	71.9	76.9
Joshi et al. (2020)	85.8	84.8	85.3	78.3	77.9	78.1	76.4	74.2	75.3	79.6
Xu and Choi	85.7	85.3	85.5	78.6	78.6	78.6	76.8	74.8	75.8	79.9
Joshi et al. (S)	86.6	84.5	85.6	80.4	77.3	78.8	77.8	74.0	75.8	80.1
Ours	86.1	85.5	**85.8**	79.8	78.8	**79.3**	77.4	75.4	**76.4**	**80.5**

Table 1: Results on the CoNLL-2012 English shared task test set. Avg. F1 in the last column denotes the average F1 of MUC, B^3, and $CEAF_{\phi_4}$. Joshi et al. (S) refers to the original end-to-end model with SpanBERT and trained with speaker encoding. **The improvements shown in the table are significant under the two-tailed *t*-test.**

	Avg. P	Avg. R	Avg. F1
Bamman et al.	-	-	68.1
LB-MEM	-	-	75.7
U-MEM	-	-	75.9
Joshi et al.	78.7	72.9	75.6
Ours	77.4	75.3	**76.3**

Table 2: Results on the test set of LitBank. The results are averaged over 10 train/dev/test splits. LB-MEM and U-MEM are reported in (Toshniwal et al., 2020).

	Avg. P	Avg. R	Avg. F1
Joshi et al. (S)	81.6	78.6	80.1
Sigmoid	80.8	79.6	80.2
MTL	80.8	78.9	79.8
Ours	81.1	79.9	**80.5**

Table 3: Comparison with three constructed baselines.

mentions. Tab. 4 shows the recall rate of mentions of different nested depth. Here, nested depth refers to the level of nesting in the mentions. E.g. in the first example given in §2.4, *The president* is of depth 1, while *he and his wife, now a New York senator* is of depth 2.

As shown in Tab. 4, the gains of our method are larger for deeply nested mentions, which highlights the capability of our structured span detector to handle harder nested mentions that cannot be handled by the greedy selector.

Nested Depth	1	2	3+
Greedy	96.5	87.1	86.2
Ours	97.8	93.2	93.9

Table 4: Recall rate of mentions of different nested depth on CoNLL-2012 dev set. There are 16873, 2100, and 182 mentions respectively of each nested depth.

Widths of Mentions. We also compare the recall rate for mentions of different widths in Tab. 5. We show that our model can detect longer spans better, which are usually more difficult to detect. For spans with 5-12 words, our structured model still maintains a recall rate of 96.5%, compared to a sharp drop of the greedy unstructured model.

Span Width	1-4	5-12	12+
Greedy	96.2	92.7	82.5
Ours	97.8	96.5	85.2

Table 5: Recall rate of mentions of different width on CoNLL-2012 dev set. There are 16356, 2180, and 619 mentions respectively of each width interval.

5.4 Semantic Role Labeling

For semantic role labeling, we report the precision, recall, and F1 score on the CoNLL-2012 SRL dataset. The gold predicates are provided during both training and evaluation. Therefore, the model have to focus on extracting the correct arguments and classifying their roles for each predicate. The

results are shown in Tab. 6 in comparison with previous span-based models. He et al.$_{\text{SpanBERT}}$ refers to the model of He et al. (2018) with SpanBERT$_{\text{large}}$ (Joshi et al., 2020) as sentence encoder.

5.4.1 Analysis of Argument Detection

	Avg. P	Avg. R	Avg. F1
He et al. (2018)	-	-	85.5
Ouchi et al. (2018)	87.1	85.3	86.2
Li et al. (2019)	85.7	86.3	86.0
Shi and Lin (2019)	85.9	87.0	86.5
He et al.$_{\text{SpanBERT}}$	88.3	85.9	87.1
`Ours`	88.1	86.9	**87.5**

Table 6: Results on the test set of CoNLL-2012 semantic role labeling task. The precision, recall, and F1 scores are averaged over all semantic roles.

Next, we report the performance of our span selector on the SRL task. Following the same trend as coreference resolution, we find that our structured model is able to extract much more accurate arguments and thus, significantly reduce the memory consumption for the downstream task. While keeping a comparable recall rate of gold arguments (96.5% for the greedy selector and 96.2% for `ours`), our span selector reduces the number of enumerated spans by 21.2%.

	P	R	F1
`BIO`	90.2	91.0	90.6
`Greedy`	92.8	90.2	91.5
`Ours`	92.8	91.2	92.0

Table 7: Comparison of unlabeled span accuracy.

We compare the accuracy of retrieving unlabeled argument spans in Tab. 7. `BIO` refers to the tagger-style SRL model using the same text encoder. Our model outperforms the other two baselines.

5.5 Qualitative Examples

In this section, we show a qualitative example to illustrate the grammar learned by our span selector for coreference resolution and SRL.

Two sets of extracted spans for the same input sentence are shown in Tab. 8. For coreference resolution, our model selects maximal NPs (containing all modifiers) and verbs. While in SRL, the parse tree consists of much denser and syntactically heterogeneous spans of NPs, PPs, modal verbs, and adverbs, etc.. This comparison empirically shows that our model is capable and robust enough to learn much more complex underlying grammars from partial annotation.

Input	The world' s fifth Disney park will soon open to the public here .
Coref	[[The world' s] fifth [Disney] park] will soon [open] to [the public] here .
SRL	[The world' s [fifth] [Disney] [park]] [will] [soon] [[open] [to the public]] [here] .

Table 8: A qualitative example of the grammar learned by the structured span selector on coreference resolution and SRL.

	Coref	SRL
`Greedy`	11.5	12.2
`Ours`	8.5	9.3

Table 9: Comparison of peak GPU memory usage in GBs at inference time on development set.

5.6 Memory Efficiency

We further analyze the memory efficiency of our model. We evaluate the peak GPU memory usage on the development set of OntoNotes. For both tasks, we see a significant reduction in memory usage of 27% for coreference and 24% for SRL.

6 Conclusion and Future Work

In this paper, we proposed a novel structured model for span selection. In contrast to prior span selection methods, the model is structured, which allows it to model the spans better. Instead of a greedy span selection procedure, the span selector uses partial span annotations provided in data to directly obtain the set of optimal spans. We evaluated our span selector on two typical span prediction tasks, namely coreference resolution and semantic role labeling, and achieved consistent gains in terms of accuracy as well as efficiency. In the future, we plan to generalize the proposed span selection model to other NLP tasks, e.g. extractive question answering (Lee et al., 2016). It would also be interesting to experiment with various pretraining techniques for span selection.

Ethical Considerations

To the best of our knowledge, the datasets used in our work do not contain sensitive information. Although our model has been evaluated under academic settings, it could also be applied in sensitive contexts, e.g. healthcare or legal scenarios. It is essential that necessary anonymization and robustness evaluation is undertaken before using our model in these settings.

References

David Bamman, Olivia Lewke, and Anya Mansoor. 2020. An annotated dataset of coreference in English literature. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 44–54, Marseille, France. European Language Resources Association.

Jason Eisner. 2016. Inside-outside and forward-backward algorithms are just backprop (tutorial paper). In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 1–17, Austin, TX. Association for Computational Linguistics.

Hongliang Fei, Xu Li, Dingcheng Li, and Ping Li. 2019. End-to-end deep reinforcement learning based coreference resolution. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 660–665, Florence, Italy. Association for Computational Linguistics.

Jenny Rose Finkel and Christopher D. Manning. 2009. Joint parsing and named entity recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 326–334, Boulder, Colorado. Association for Computational Linguistics.

David Gaddy, Mitchell Stern, and Dan Klein. 2018. What's going on in neural constituency parsers? An analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 999–1010, New Orleans, Louisiana. Association for Computational Linguistics.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018. Jointly predicting predicates and arguments in neural semantic role labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 364–369, Melbourne, Australia. Association for Computational Linguistics.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 473–483, Vancouver, Canada. Association for Computational Linguistics.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Mandar Joshi, Omer Levy, Luke Zettlemoyer, and Daniel Weld. 2019. BERT for coreference resolution: Baselines and analysis. 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 5803–5808, Hong Kong, China. Association for Computational Linguistics.

Daniel Jurafsky and James H. Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 1st edition. Prentice Hall PTR, USA.

Ben Kantor and Amir Globerson. 2019. Coreference resolution with entity equalization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 673–677, Florence, Italy. 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.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 687–692, New Orleans, Louisiana. Association for Computational Linguistics.

Kenton Lee, Shimi Salant, Tom Kwiatkowski, Ankur Parikh, Dipanjan Das, and Jonathan Berant. 2016. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436*.

Zuchao Li, Shexia He, Hai Zhao, Yiqing Zhang, Zhuosheng Zhang, Xi Zhou, and Xiang Zhou. 2019. Dependency or span, end-to-end uniform semantic role labeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6730–6737.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 411–420, Vancouver, Canada. Association for Computational Linguistics.

Lesly Miculicich and James Henderson. 2019. Partially-supervised mention detection. *CoRR*, abs/1908.09507.

Hiroki Ouchi, Hiroyuki Shindo, and Yuji Matsumoto. 2018. A span selection model for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1630–1642, Brussels, Belgium. Association for Computational Linguistics.

Martha Palmer, Daniel Gildea, and Nianwen Xue. 2010. Semantic role labeling. *Synthesis Lectures on Human Language Technologies*, 3(1):1–103.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40, Jeju Island, Korea. Association for Computational Linguistics.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A multi-pass sieve for coreference resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 492–501, Cambridge, MA. Association for Computational Linguistics.

Peng Shi and Jimmy Lin. 2019. Simple BERT models for relation extraction and semantic role labeling. *CoRR*, abs/1904.05255.

Noah A. Smith. 2011. Linguistic structure prediction. *Synthesis Lectures on Human Language Technologies*, 4(2):1–274.

Manfred Stede. 2011. Discourse processing. *Synthesis Lectures on Human Language Technologies*, 4(3):1–165.

Swabha Swayamdipta, Sam Thomson, Kenton Lee, Luke Zettlemoyer, Chris Dyer, and Noah A. Smith. 2018. Syntactic scaffolds for semantic structures. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3772–3782, Brussels, Belgium. Association for Computational Linguistics.

Shubham Toshniwal, Sam Wiseman, Allyson Ettinger, Karen Livescu, and Kevin Gimpel. 2020. Learning to Ignore: Long Document Coreference with Bounded Memory Neural Networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8519–8526, Online. Association for Computational Linguistics.

Wei Wu, Fei Wang, Arianna Yuan, Fei Wu, and Jiwei Li. 2020. CorefQA: Coreference resolution as query-based span prediction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6953–6963, Online. Association for Computational Linguistics.

Liyan Xu and Jinho D. Choi. 2020. Revealing the myth of higher-order inference in coreference resolution. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8527–8533, Online. Association for Computational Linguistics.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2019. Neural mention detection. *CoRR*, abs/1907.12524.

Rui Zhang, Cícero Nogueira dos Santos, Michihiro Yasunaga, Bing Xiang, and Dragomir Radev. 2018. Neural coreference resolution with deep biaffine attention by joint mention detection and mention clustering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 102–107, Melbourne, Australia. Association for Computational Linguistics.

Yu Zhang, Qingrong Xia, Shilin Zhou, Yong Jiang, Zhenghua Li, Guohong Fu, and Min Zhang. 2021. Semantic role labeling as dependency parsing: Exploring latent tree structures inside arguments.

A A Weighted Context-Free Grammar

In our WCFG $\langle \Sigma, N, S, R \rangle$, there are only three types of nonterminals, i.e. $N = \{S, \sigma, \bar{\sigma}\}$, where S is the start symbol, σ is the span of interest, and $\bar{\sigma}$ is the spans that are not of interest.

Consequently, there are very few production rules in R. And we only assign nontrivial weights to rules $A \rightarrow BC$ where at least one of B, C is σ. That is to say, in Eq. (1), $s_p(r) = 1$ for all $r = A \rightarrow BC$ where neither B nor C is σ. This grammar is described in (Gaddy et al., 2018, Sec. 2.5) to eliminate the need for binarization for n-ary trees.

B Experimental Settings

The systems are implemented with PyTorch. We use SpanBERT$_{\text{large}}$ as text encoder. We train the model for 20 epochs and select the best performing model on the development set for testing. The documents are split into 512 word segments to fit in SpanBERT$_{\text{large}}$. Models used for coreference resolution have 402 million learnable parameters, and models for SRL have 382 million learnable parameters.

We closely follow the hyperparameter settings of Joshi et al. (2020) and build our models upon the codebase of Xu and Choi (2020) [7] under Apache License 2.0. The learning rate of SpanBERT$_{\text{large}}$ parameters is set to 1×10^{-5} with 0.01 decay rate, and the learning rate of task parameters is set to 3×10^{-4}. The dropout rate of feedforward neural network scorers is set to 0.3. When training our model, we randomly sample $0.1|D|$ negative spans that are not mentions and add their negative log-likelihood $-\log p(\sigma_{ij})$ to the training objective. This is to prevent $p(\sigma_{ij})$ from converging to 1.

For SRL task, we use a batch size of 32 for 40 epochs and the same learning rate with coreference resolution. The same negative sampling technique is applied.

Our models are trained on Nvidia Tesla V100 GPUs with 32GB memory. The average training time is around 8 hours for Joshi et al. (S) baseline, and around 9 hours for our model. For SRL models, training takes 25 hours.

C Results on the Development Set

In this section, we report the results that our models get on the development sets of OntoNotes and LitBank.

	Avg. P	Avg. R	Avg. F1
Joshi et al. (S)	82.0	78.8	80.4
`Sigmoid`	81.6	79.4	80.5
`Ours`	81.1	80.2	**80.7**

Table 10: Results on CoNLL-2012 coreference resolution development set.

	Avg. P	Avg. R	Avg. F1
Joshi et al. (S)	78.8	73.7	76.1
`Ours`	77.4	75.6	**76.6**

Table 11: Results on LitBank development set.

	Avg. P	Avg. R	Avg. F1
He et al.$_{\text{SpanBERT}}$	87.9	85.0	86.4
`Ours`	87.8	86.2	**87.0**

Table 12: Results on CoNLL-2012 semantic role labeling development set.

[7] https://github.com/lxucs/coref-hoi