Sentence-Incremental Neural Coreference Resolution

Matt Grenander   **Shay B. Cohen**   **Mark Steedman**
School of Informatics
University of Edinburgh
matt.grenander@ed.ac.uk
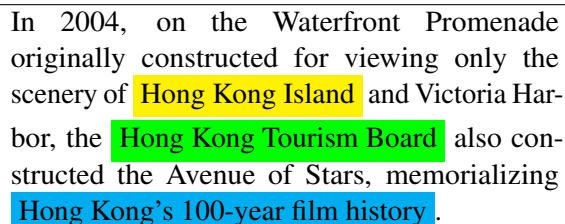scohen@inf.ed.ac.uk, m.steedman@ed.ac.uk

Abstract

We propose a sentence-incremental neural coreference resolution system which incrementally builds clusters after marking mention boundaries in a shift-reduce method. The system is aimed at bridging two recent approaches at coreference resolution: (1) state-of-the-art non-incremental models that incur quadratic complexity in document length with high computational cost, and (2) memory network-based models which operate incrementally but do not generalize beyond pronouns. For comparison, we simulate an incremental setting by constraining non-incremental systems to form partial coreference chains before observing new sentences. In this setting, our system outperforms comparable state-of-the-art methods by 2 F1 on OntoNotes and 6.8 F1 on the CODI-CRAC 2021 corpus. In a conventional coreference setup, our system achieves 76.3 F1 on OntoNotes and 45.5 F1 on CODI-CRAC 2021, which is comparable to state-of-the-art baselines. We also analyze variations of our system and show that the degree of incrementality in the encoder has a surprisingly large effect on the resulting performance.[1]

1 Introduction

Coreference Resolution (CR) is a task in which a system detects and resolves linguistic expressions that refer to the same entity. It is typically performed in two steps: in *mention detection*, the model predicts which expressions are referential, and in *mention clustering*, the model computes each mention's antecedent. Many recently proposed systems follow a mention-pair formulation from Lee et al. (2017), in which all possible spans are ranked and then scored against each other. In particular, methods that augment this approach with large, pre-trained language models achieve state-of-the-art results (Joshi et al., 2019, 2020).

---

[1]Code is available at:
https://github.com/mgrenander/sentence-incremental-coref

Figure 1: An example from the OntoNotes dataset which highlights the need for incremental systems to identify spans rather than tokens as mentions. The mentions cannot be resolved solely from the prefix 'Hong Kong', and the clustering decision should be delayed until the full mention is observed.

Despite impressive performance, these methods are computationally demanding. For a text with n tokens, they will score up to $O(n^2)$ spans, followed by up to $O(n^4)$ span comparisons. They also process documents non-incrementally, requiring access to the entire document before processing can begin. These properties present challenges when insufficient computational resources are available, or when the task setup is incremental, such as in dialogue (e.g. Khosla et al. 2021). From a cognitive perspective, these methods are also unappealing because research on "garden-path" effects show that humans resolve referring expressions incrementally (Altmann and Steedman, 1988).

These drawbacks have led to renewed interest in incremental coreference resolution systems, in which document tokens are processed sequentially. Some recent approaches use memory networks to track entities in differentiable memory cells (Liu et al., 2019; Toshniwal et al., 2020a). These models demonstrate proficiency at proper name and pronoun resolution (Webster et al., 2018). However, they seem unlikely to generalize to more complicated coreference tasks due to a strict interpretation of incrementality. Both Liu et al. (2019) and Toshniwal et al. (2020a) resolve mentions word-by-word, making coreference decisions possibly

before the full noun phrase has been observed. The approach is adequate for proper names and pronouns, but it may fail to distinguish entities who share the same phrase prefix. For example, in Figure 1, three mentions all begin with 'Hong Kong', though all belong to separate entities. In this case, it is difficult to see how a system using word-level predictions would resolve these mentions to different entities.

Motivated by this recent work, we propose a new system that processes a document incrementally at the sentence-level, creating and updating coreference clusters after each sentence is observed. The system addresses deficiencies in memory network-based approaches by delaying mention clustering decisions until the full mention has been observed. These goals are achieved through a novel mention detector based on shift-reduce parsing, which identifies mentions by marking left and right mention boundaries. Identified mention candidates are then passed to an online mention clustering model similar to Toshniwal et al. (2020b) and Xia et al. (2020). The model proposes a linear number of spans per sentence, reducing computational requirements and maintaining more cognitive plausibility compared to non-incremental methods.

In order to compare non-incremental and incremental systems on equal footing, we propose a new **sentence-incremental** evaluation setting. In this setting, systems receive sentences incrementally and must form partial coreference chains before observing the next sentence. This setting mimics human coreference processing more closely, and is a more suitable evaluation setting for downstream tasks in which full document access is generally not available (e.g. for dialogue (Andreas et al., 2020)).

Using the sentence-incremental setting, we demonstrate that our model outperforms comparable systems adapted from partly incremental methods (Xia et al., 2020) across two corpora, the OntoNotes dataset (Pradhan et al., 2012) and the recently released CODI-CRAC 2021 corpus (Khosla et al., 2021). Moreover, we show that in a conventional evaluation setting, where the model can access the entire document, our system retains close to state-of-the-art performance. However, the sentence-incremental setting is substantially outperformed by non-sentence-incremental systems. Analyzing the difference between these two settings reveals that the encoder is heavily dependent on how many sentences it can observe at a time.

The analysis suggests better representations of the entities and their context may improve performance in the sentence-incremental setting. Nevertheless, our results provides new state-of-the-art baselines for sentence-incremental evaluation.

2 Related Work

Non-incremental mention-pair models have dominated the field in recent years, with many following the formulation presented by Lee et al. (2017). Several extensions have led to performance improvements, such as adding higher-order inference (Lee et al., 2018), and replacing the encoder with BERT and SpanBERT (Joshi et al., 2019, 2020). Extensions to this approach have looked at reformulating the problem as question-answering (Wu et al., 2020), simplifying span representations (Kirstain et al., 2021), and incorporating coherence signals from centering theory (Chai and Strube, 2022). Although our work is orthogonal to this line of research, we compare our system against this type of non-incremental model.

Toshniwal et al. (2020b) and Xia et al. (2020) adapt the non-incremental system of Joshi et al. (2020) so that mention clustering is performed incrementally. Their resulting models achieve similar performance to the original non-incremental one. However, in their formulation, document encoding, mention detection and certain clustering decisions still fully depend on Joshi et al. (2020). The resulting model still requires access to the full document in order to compute coreference chains. Yu et al. (2020b) similarly present an incremental mention clustering approach where mention detection is performed non-incrementally as Lee et al. (2017).

Memory network-based approaches identify co-referring expressions by writing and updating entities into cells within a fixed-length memory (Liu et al., 2019; Toshniwal et al., 2020a). These models demonstrate how fully incremental coreference systems can be achieved. However, the formulation operates on token-level predictions, and does not easily extend to either nested mentions or certain multi-token mentions (e.g. in Figure 1).

Cross-document coreference resolution (CDCR) requires systems to compute coreference chains across documents, raising scalability challenges as the number of documents increases. Given these challenges, incremental CDCR systems are crucial (Allaway et al., 2021; Logan IV et al., 2021) due to lower memory requirements. However, these

works are not directly comparable to ours since they assume gold mentions are provided as input.

Other, earlier, incremental coreference systems also often ignore or diminish the role of mention detection. For example, Webster and Curran (2014) use an external parser for mention detection, requiring an additional model. Klenner and Tuggener (2011) assume gold mentions as input.

Recently, Liu et al. (2022) propose a coreference resolution system based a seq2seq formulation using hidden variables. Although their focus is on adding structure to seq2seq models, their system can also be viewed as transition-based like ours.

Our incremental mention detector bears similarities to certain models for nested named-entity recognition (NER). In particular, Wang et al. (2018) present an incremental neural model for nested NER based on a shift-reduce algorithm. Their deduction rules differ greatly from ours as they model mention spans using complete binary trees, and are aimed at NER rather than mention detection.

Recent work has also explored incremental transformer architectures (Katharopoulos et al., 2020; Kasai et al., 2021), and adapting these architectures to NLU tasks (though not coreference resolution) (Madureira and Schlangen, 2020; Kahardipraja et al., 2021). In this work, we focus on the simpler sentence-incremental setting, believing it to be sufficient for downstream tasks.

3 Method

Given a document, the goal is to output a set of clusters $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_K\}$, where mentions within each cluster are co-referring. We assume mentions may be nested but otherwise do not overlap. This assumption allows us to model mentions using a method analogous to shift-reduce, where shifting corresponds to either incrementing the buffer index or marking a left mention boundary, and reducing corresponds to marking a right boundary and resolving the mention to an entity cluster.

3.1 Shift-Reduce Framework

The main idea is to mark mention boundaries using PUSH, POP or PEEK actions, or to pass over a non-boundary token with the ADVANCE action. After POP or PEEK actions, a mention candidate is created using the current top-of-stack and buffer elements. The resulting mention candidate is then either resolved to an existing cluster or initialized as a new entity cluster.

$$
\begin{array}{ll}
\text{Initial} & [\varnothing, 0, \varnothing, \varnothing] \\
\text{Final} & [\varnothing, n, A, \mathcal{C}]
\end{array}
$$

$$
\text{PUSH} \quad \frac{[S, i, A, \mathcal{C}]}{[S|w_i, i, A|\text{PUSH}, \mathcal{C}]}
$$

$$
\text{ADVANCE} \quad \frac{[S, i, A, \mathcal{C}]}{[S, i+1, A|\text{ADVANCE}, \mathcal{C}]}
$$

$$
\text{POP} \quad \frac{[S|v, i, A, \mathcal{C}]}{[S, i, A|\text{POP}, \mathcal{C}|\text{COREF}(v, w_i)]}
$$

$$
\text{PEEK} \quad \frac{[S|v, i, A, \mathcal{C}]}{[S|v, i, A|\text{PEEK}, \mathcal{C}|\text{COREF}(v, w_i)]}
$$

Figure 2: Deduction rules for our coreference resolver. $[S, i, A, \mathcal{C}]$ denotes the stack S, buffer index i, action history A, and cluster set $\mathcal{C}$. The COREF function indicates that span (v, w_i) is clustered and added to $\mathcal{C}$.

We represent the state as $[S, i, A, \mathcal{C}]$, where S is the stack, i is the buffer index, A is the action history and $\mathcal{C}$ is the current set of clusters. At each time step, one of four actions is taken:

- PUSH: Place the word at buffer index i on top of the stack, marking a left mention boundary.

- ADVANCE: Move the buffer index forward.

- POP: Remove the top element from S and create a mention candidate using this element and the current buffer element. Score the candidate against existing clusters and resolve it (or create a new cluster).

- PEEK: Create a mention candidate using the top element on the stack and the current buffer element. Score the candidate against existing clusters and resolve it (or create a new cluster).

The PEEK action does not alter the stack but is otherwise identical to POP. This action is critical for detecting mentions sharing a left boundary.

Several hard action constraints ensure that only valid actions are taken and the final state is always reached. For example, PUSH can only be called once per token, or else the model would be marking the left boundary multiple times. The full list of constraints is described in the appendix.

We denote the set of valid actions as $\mathcal{V}(S, i, A, \mathcal{C})$. The conditional probability of selecting action a_t based on state $\mathbf{p}_t$ can then be expressed as:

$$p_M(a_t | \mathbf{p}_t) = \frac{\exp(w_{a_t} \cdot f_M(\mathbf{p}_t))}{\sum_{a' \in \mathcal{V}(S,i,A,\mathcal{C})} \exp(w_{a'} \cdot f_M(\mathbf{p}_t))},$$

where f_M is a two-layer neural network, and w_{a_t} is a column vector selecting action a_t.

If POP or PEEK operations are predicted, the mention candidate is then scored against existing clusters. Depending on these scores, the mention is either (a) resolved to an existing cluster, or (b) initialized as a new entity cluster. Define the set of possible coreference actions as $\mathcal{A}_k$, which includes resolving to existing clusters $\{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ and creating new cluster $\mathcal{C}_{k+1}$. We can write the conditional probability of coreference prediction z_j based on mention candidate m_j as:

$$p_C(z_j | m_j) = \frac{\exp(w_{z_j} \cdot s_C(m_j))}{\sum_{z' \in \mathcal{A}_k} \exp(w_{z'} \cdot s_C(m_j))},$$

where s_C is a function scoring the mention candidate against $\{\mathcal{C}_1, \ldots, \mathcal{C}_k, \mathcal{C}_{k+1}\}$ (described in Section 3.2.2).

The terminal state is reached when the final buffer element has been processed and the stack is empty. At this point, all mentions have been clustered and we return all non-singleton entity clusters. Figure 2 presents a more formal description of the deduction rules, while an example is also shown in Figure 3.

3.2 Neural Implementation

3.2.1 Mention Detector

Document tokens are first encoded using a pre-trained language model. The concatenated word embeddings, $x_1, \ldots, x_n$, form the **buffer** for the shift-reduce mechanism. Assuming current word x_i and time step t, we denote the buffer as $\mathbf{b}_t = x_i$.

The stack is represented using a Stack-LSTM (Dyer et al., 2015). Let $x_{s_1}, \ldots, x_{s_L}$ be the currently marked left mention boundaries pushed to the stack. Then the **stack** representation at time t is:

$$\mathbf{s}_t = \text{StackLSTM}[x_{s_1}, \ldots, x_{s_L}].$$

We encode the action history $a_0, \ldots, a_{t-1}$ with learned embeddings for each of the four actions.

The action history at t is encoded with an LSTM over previous action embeddings:

$$\mathbf{a}_t = \text{LSTM}[a_0, \ldots, a_{t-1}].$$

Then, the parser state is represented by the concatenation of buffer, stack, action history and additional mention features ϕ_M:

$$\mathbf{p}_t = [\mathbf{b}_t; \mathbf{s}_t; \mathbf{a}_t; \phi_M(\mathbf{b}_t, \mathbf{s}_t)],$$

where ϕ_M denotes learnable embeddings corresponding to useful mention features such as span width and document genre. For span width, we use embeddings measuring the distance from the top of the stack to the current buffer token (i.e. $i - s_L$), or 0 if the stack is empty.

3.2.2 Mention Clustering Model

The mention clustering is similar to previous online clustering methods (Toshniwal et al., 2020b; Xia et al., 2020; Webster and Curran, 2014), though we take care to avoid dependence on non-incremental pre-trained language models which have already been fine-tuned to this task.

Given a mention candidate's span representation v, we score v against the existing entity cluster representations $\mathbf{m}_1, \ldots, \mathbf{m}_k$:

$$s_C(v) = [f_C(\mathbf{m}_1, v), \ldots, f_C(\mathbf{m}_k, v), \alpha]$$
$$f_C(\mathbf{m}_i, v) = \text{MLP}([v, \mathbf{m}_i, v \odot \mathbf{m}_i, \phi_C(v, \mathbf{m}_i)])$$
$$i^* = \underset{i \in \{1, \ldots, k+1\}}{\arg\max} \; s_C(v)$$

where f_C is two-layer neural network, α is a threshold value for creating a new cluster, $v \odot \mathbf{m}_i$ is the element-wise product and ϕ_C encodes useful features between v and $\mathbf{m}_i$: the number of entities in $\mathbf{m}_i$, mention distance between v and $\mathbf{m}_i$, the previous coreference action and document genre.

If the scores between v and all cluster representations $\mathbf{m}_1, \ldots, \mathbf{m}_k$ are below some threshold value α (i.e. $i^* = k + 1$), we initialize a new entity cluster with v. Otherwise, we update the cluster representation $\mathbf{m}_{i^*}$ via a weighted average using the number of entities represented by $\mathbf{m}_{i^*}$:

$$\mathbf{m}_{i^*} \leftarrow \beta \cdot \mathbf{m}_{i^*} + (1 - \beta) \cdot v,$$

where $\beta = \frac{|\mathbf{m}_{i^*}|}{|\mathbf{m}_{i^*}| + 1}$ is the weighting term.

Step	Action(s)	Stack	Buffer	Clusters
1.	PUSH	[Auto]	Auto workers ended their strike	∅
2.	ADVANCE	[Auto]	workers ended their strike	∅
3.	POP	[∅]	workers ended their strike	{ Auto workers }
4.	ADVANCE	[∅]	ended their strike	{ Auto workers }
5.	ADVANCE	[∅]	their strike	{ Auto workers }
6.	PUSH	[their]	their strike	{ Auto workers }
7.	PEEK	[their]	their strike	{ Auto workers, their }
8.	ADVANCE	[their]	strike	{ Auto workers, their }
9.	POP	[∅]	strike	{ Auto workers, their } { their strike }
10.	ADVANCE	[∅]	∅	{ Auto workers, their } { their strike }

Figure 3: Example of the shift-reduce system for the sentence "Auto workers ended their strike". ∅ denotes the empty stack or empty cluster set. Expressions within brackets mean they are co-referring. In each step, the Stack and Buffer show the result of applying the given action.

3.2.3 Training

Training is done via teaching forcing. At each time step, the model predicts the gold action given the present state. The state is then updated using the gold action. At each step, we compute mention detection loss $\mathcal{L}_M$ and coreference loss $\mathcal{L}_C$.

The mention detection loss $\mathcal{L}_M$ is calculated using the cross-entropy between the predicted mention detection action and gold action $a_{t^*} \in \mathcal{V}(S, i, A, \mathcal{C})$:

$$\mathcal{L}_M = -\sum_t \log p_M(a_{t^*} \mid \mathbf{p}_t),$$

where t sums over time steps across all documents.

Similarly, the coreference loss $\mathcal{L}_C$ is defined by the cross entropy between the highest-scoring coreference action and the gold coreference decision $z_{j^*} \in \mathcal{A}_k$:

$$\mathcal{L}_C = -\sum_j \log p_C(z_{j^*} \mid m_j),$$

where j sums over mentions across all documents. The entire network is then trained to optimize the sum of the two losses, $\mathcal{L}_M + \mathcal{L}_C$. During inference, we predict actions using greedy decoding, updating the state solely with predicted actions.

Figure 4 presents a summary of the various components and the overall algorithm.

4 Experiments

4.1 Datasets

We train and evaluate our system on the **OntoNotes 5.0** dataset (Weischedel et al., 2013), using the same setup described in the CoNLL-2012 Shared Task (Pradhan et al., 2012). OntoNotes includes 7 document genres and does not restrict mention token length; annotations cover pronouns, noun phrases and heads of verb phrases. We evaluate using the *MUC* (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998) and $CEAF_{\phi_4}$ (Luo, 2005) metrics and their average (the CoNLL score), using the official CoNLL-2012 scorer.

We also test models on the the recently released **CODI-CRAC 2021** corpus (Khosla et al., 2021). This dataset annotates coreference (and other anaphora-related tasks) for 134 documents across 4 separate dialogue corpora (Light, Urbanek et al. 2019, AMI, Carletta 2006, Persuasion, Wang et al. 2019 and Switchboard, Godfrey et al. 1992). The dataset suits incremental systems well since dialogue can be naturally presented as incremental utterances. Given the small dataset size, we use it for evaluation only, using models trained on OntoNotes. Since OntoNotes marks document genre (which systems often use as a feature), we associate CODI-CRAC documents with OntoNotes' 'telephone conversation' genre, since it is the most similar. We remove singleton clusters due to lack of annotation in the training set. We again evaluate using *MUC*, B^3 and $CEAF_{\phi_4}$, using the official Universal Anaphora scorer (Yu et al., 2022).

4.2 Model Components

4.2.1 Document Encoder

Recent models on coreference resolution often use SpanBERT (Joshi et al., 2020) for word embeddings (Wu et al. (2020); Toshniwal et al. (2020b); Xia et al. (2020); Xu and Choi (2020), among others), since Joshi et al. (2020) demonstrate Span-
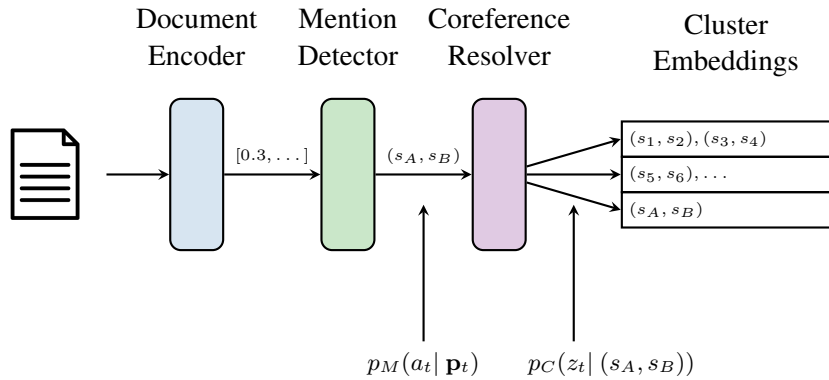
Figure 4: A summary of the overall algorithm. After document encoding, the mention detector predicts transition actions PUSH, POP, PEEK or ADVANCE using the parser state $\mathbf{p}_t$. If a mention is predicted, the coreference resolver then clusters it to an existing cluster representation or creates a new cluster. Clustering a mention implies a coreference relation with mentions in the cluster. The steps can all be performed incrementally, assuming the document encoder is also incremental.

BERT's proficiency for entity-related tasks such as coreference resolution. However, SpanBERT is unsuitable for incremental applications because it expects all its input simultaneously and cannot partially process text while waiting for future input.

Instead, we turn to XLNet[2] (Yang et al., 2019), which extends the earlier Transformer-XL (Dai et al., 2019). XLNet differs from typical pre-trained language models as it can efficiently cache and reuse its previous outputs. The caching mechanism allows for recurrent computation to be performed efficiently. Cached outputs provide a context to the current sentence being processed.

We experiment using XLNet in two settings: in the **Sentence-Incremental** (Sent-Inc) setting, each sentence is processed sequentially, and partial coreference clusters are computed before the next sentence is observed. After each sentence is processed, we accumulate XLNet's outputs (up to a cutoff point) and reuse them when processing the next sentence. We limit the number of cached tokens so that the cached and 'active' tokens do not exceed 512, so that our work remains comparable to other recent works. Although the mention detector is token-incremental and the mention clustering component is span-incremental, the document encoder is sentence-incremental, so overall we describe the system as sentence-incremental.

In the **Part-Incremental** (Part-Inc) setting, we allow XLNet to access multiple sentences simultaneously. We experiment both with and without the cache mechanism, using up to a total of 512 tokens at a time. This setting is comparable to experiments

in Xia et al. (2020) and Toshniwal et al. (2020b), where document encoding is also non-incremental. In our case, both mention detection and mention clustering components remain incremental as in the Sentence-Incremental setting. In this way, we can isolate the effect of sentence-incrementality on the document encoder (XLNet).

4.2.2 Span Representation

We use a similar span representation to Lee et al. (2017): for a span (i, j), we concatenate word embeddings (x_i, x_j), an attention-weighted average $\bar{x}$ and learnable embeddings for span width and speaker ID (the speaker for (i, j)). We use 20-dimensional learned embeddings for all features (span width, speaker ID, document genre, action history, mention distance and number of entities in each cluster).

4.2.3 Training

We use Adam to train task-specific parameters, and AdamW for XLNet's parameters (Kingma and Ba, 2015; Loshchilov and Hutter, 2019). The gradient is accumulated across one document before updating model weights. We use a learning rate scheduler with a linear decay, and additionally warmup SpanBERT's parameters for the first 10% update steps. For the mention detector, we balance the loss weights based on the frequency of each action in the training set. This step is important because most tokens do not correspond to mention boundaries, meaning the ADVANCE action is by far the most prevalent in the training set.

Training converges within 15 epochs. The model is implemented in PyTorch (Paszke et al., 2019). A

---

[2]We use the *base* version due to memory restrictions.

complete list of hyperparameters is included in the appendix.

4.3 Comparisons

We compare against several recent works with varying degrees of incrementality. Table 1 summarizes their differences in incrementality compared to ours, as well as the span complexity. Joshi et al. (2020) is a non-incremental formulation: it adopts the *end-to-end* and *coarse-to-fine* formulations from Lee et al. (2017) and Lee et al. (2018), replacing the LSTM encoder with their novel Span-BERT architecture.

Model	Incremental Components	Span Complexity
SpanBERT	None	$O(n^4)$
longdoc	Mention Clustering	$O(n^2m)$
ICoref	Mention Clustering	$O(n^2m)$
Part-Inc (Ours)	Mention Detection + Mention Clustering	$O(nm)$
ICoref-*inc*	All	$O(n^2m)$
Sent-Inc (Ours)	All	$O(nm)$

Table 1: The list of systems we compare, alongside their incrementality (on a sentence-level) and span complexity. 'All Components' means document encoding, mention detection and mention clustering. n is the number of tokens and m is the number of entities.

The longdoc (Toshniwal et al., 2020b) and ICoref (Xia et al., 2020) systems adapt Joshi et al. (2020) so that mention clustering is done incrementally. However, both models avoid modifying the non-incremental document encoding and mention detection steps from Joshi et al. (2020), and the resulting systems are only partly incremental. Since Toshniwal et al. (2020b) and Xia et al. (2020) only experiment with SpanBERT-large, we re-train their implementations with SpanBERT-base to fairly compare against our own systems.

Xia et al. (2020) also provide a truly sentence-incremental version of their system, which we call ICoref-*inc*[3]. This version is trained by encoding tokens and proposing mentions sentence-by-sentence, independently processing each sentence as it is observed while maintaining entity clusters across sentences. Since ICoref-*inc* is fully sentence-incremental, it provides the fairest comparison to our own Sentence-Incremental setting. Having more incremental components results in

---

[3]Specifically, this system is the "Train 1-sentence / Inference 1-sentence" model from Xia et al. (2020)'s Table 4.

increased difficulty on the coreference task, as the system must rely on partial information when making clustering decisions.

We do not compare against Liu et al. (2019) and Toshniwal et al. (2020a)'s token-incremental models. Besides being generally unsuitable for span-based coreference, they also do not handle nested mentions. Roughly 11% of OntoNotes' mentions are nested, meaning that training these systems on OntoNotes is infeasible.

4.3.1 Span Complexity

Table 1 also compares the *span complexity* between systems, in terms of how many spans must be scored and compared. This comparison is analytic and not runtime-based, and so ignores handcrafted memory-saving techniques such as eviction and span pruning. Joshi et al. (2020) score all possible spans and compare them pairwise, meaning their system runs in $O(n^4)$, where n is the number of tokens. Toshniwal et al. (2020b) and Xia et al. (2020) reduce the complexity to $O(n^2m)$, where m is the number of entities, by incrementally clustering mentions. Finally, we claim our systems' span complexity is $O(nm)$. Our mention detector proposes $O(n)$ spans, as we can show each action is linearly bounded in the number of tokens. Our reduced complexity speaks to its increased cognitive plausibility compared to the part- and non-incremental systems, which consider a quadratic number of spans.

Note that the runtime is not comparable because non-incremental methods process the entire document in parallel, whereas ours is not parallelizable and therefore slower. We also note that this comparison does not have any bearing on memory requirements, since Toshniwal et al. (2020b) and Xia et al. (2020) both maintain constant memory through eviction strategies.

5 Results

5.1 OntoNotes

The main results for OntoNotes are shown in Table 2. First, SpanBERT (Joshi et al., 2020), being non-incremental, unsurprisingly outperforms other systems, both part and sentence incremental.

Within partly incremental systems, the ICoref model (Xia et al., 2020) performs best, below Span-BERT by 0.4 F1. Our Part-Inc model performs comparably to longdoc (Toshniwal et al., 2020b), only trailing ICoref by 0.7 F1 points.

Model	SI	MUC			B^3			$CEAF_{\phi_4}$			Avg. F1
		Rec.	Prec.	F1	Rec.	Prec.	F1	Rec.	Prec.	F1	
SpanBERT-l (Joshi et al., 2020)	None	84.8	85.8	85.3	77.9	78.3	78.1	74.2	76.4	75.3	79.6
CorefQA-l (Wu et al., 2020)	None	87.4	88.6	88.0	82.0	82.4	82.2	78.3	79.9	79.1	83.1
s2e (Kirstain et al., 2021)	None	85.1	86.5	85.8	77.9	80.3	79.1	75.4	76.8	76.1	80.3
s2e+se_ct(Chai and Strube, 2022)	None	85.3	87.2	86.3	78.6	80.7	79.6	75.2	78.2	76.7	80.9
SpanBERT-b (Joshi et al., 2020)	None	83.1	84.3	83.7	75.3	76.2	75.8	71.2	74.6	72.9	77.4
CorefQA-b (Wu et al., 2020)	None	87.4	85.2	86.3	76.5	78.7	77.6	75.6	76.0	75.8	79.9
longdoc (Toshniwal et al., 2020b)	Part	83.3	83.0	83.2	75.5	74.1	74.8	70.1	72.8	71.4	76.4
ICoref (Xia et al., 2020)	Part	83.1	84.2	83.6	74.3	75.8	75.0	71.7	73.3	72.5	77.0
Part-Inc (Ours)	Part	83.7	82.1	82.9	75.9	73.0	74.4	68.8	74.5	71.6	76.3
ICoref-*inc* (Xia et al., 2020)	All	74.0	79.7	76.7	58.6	70.6	64.0	63.7	63.1	63.4	68.0
Sent-Inc (Ours)	All	78.1	79.4	78.8	68.9	68.3	68.6	55.8	71.2	62.5	70.0

Table 2: Main results on the OntoNotes 5.0 test set with the CoNLL 2012 Shared Task metrics and the average F1 (the CoNLL F1 score). The 'SI' column denotes the sentence-incrementality of each system, summarizing details in Table 1. The top four systems are not directly comparable to ours, since they train with a 'large' encoder (either SpanBERT or Longformer (Beltagy et al., 2020)). Note that scores for Xia et al. (2020) and Toshniwal et al. (2020b) differ from their reported results because we re-train them with SpanBERT-*base* instead of *large*.

Model	LIGHT	AMI	PERS.	SWBD.	Avg.
SpanB	57.7	33.8	53.7	50.2	48.9
ICoref	54.7	33.7	51.5	48.1	47.0
Part-Inc	53.5	32.4	50.5	46.9	45.8
IC-*inc*	45.5	21.9	41.3	36.6	36.3
Sent-Inc	50.5	31.6	46.4	44.7	43.3

Table 3: Main results for the CODI-CRAC 2021 corpus. All scores denote the CoNLL F1 score (average of MUC, B^3 and $CEAF_{\phi_4}$). Here, SpanB=SpanBERT, and IC-*inc*=ICoref-*inc*.

The advantages of our method are more evident in the sentence-incremental evaluation. Since ICoref-*inc* relies on SpanBERT to encode tokens and score mentions, its performance suffers considerably when evaluated in the sentence-incremental setting. In contrast, the Sent-Inc model effectively uses the history of previous processed sentences and outperforms ICoref-*inc* by 2 F1 points. Still, both systems suffer considerably when compared to their part-incremental counterparts: ICoref drops by 9 F1 points and our model by 6.3 F1. In Section 6, we explore the main causes of this drop.

5.2 CODI-CRAC

The results on the CODI-CRAC corpus are shown in Table 3. We observe many of the same trends as in OntoNotes: the non-incremental SpanBERT again surpasses other models, achieving 2.9 F1 higher than ICoref.

Within partly incremental systems, our Part-Inc system trails ICoref by 1.2 F1. We omit the long-doc results from this table, after finding its performance surprisingly plummets when evaluated on CODI-CRAC. On all subsets, it scores below 2 F1, indicating issues with model transfer. Other works have explored this topic in depth (Toshniwal et al., 2021), and we do not investigate further here.

In the Sentence Incremental setting, although our Sent-Inc model again outperforms Xia et al. (2020)'s ICoref-*inc*, the performance difference is much larger here: 7 F1 compared to 2 F1 in OntoNotes. The gap between the Sent-Inc and Part-Inc is also much smaller: only 2.5 F1 points compared to 6.3 F1 on OntoNotes. The difference in performances between the two datasets may suggest our model is better suited to the inherent incrementality in a dialogue setting.

6 Analysis

The dramatic performance gap between the Sent-Inc and Part-Inc settings may be surprising. Since coreference resolution is primarily processed incrementally by humans, why does access to future tokens affect the Sent-Inc model so heavily?

To investigate this issue deeper, we design additional k-Sentence-Incremental settings. In each setting, the system accesses k sentences ($S_1, \ldots, S_k$) at a time as active input, and $512 - \sum_{i=1}^k |S_k|$ tokens as memory. In each setting, the model observes the same number of tokens (512), but varies the amount of active input vs. memory. The mention detection and mention clustering steps remain the same and are still incremental; the only change is in the encoder.

Varying k in this way allows us to the test the effect of more or less incrementality on the system. When $k = 1$, we recover the original Sent-Inc model. When k is large enough (in practice, 24), we get the Part-Inc model. For each $k \in \{1, 4, 8, 12, 16, 20, 24\}$, we fully train the corresponding model on OntoNotes as described in Section 4, and evaluate on the dev set.
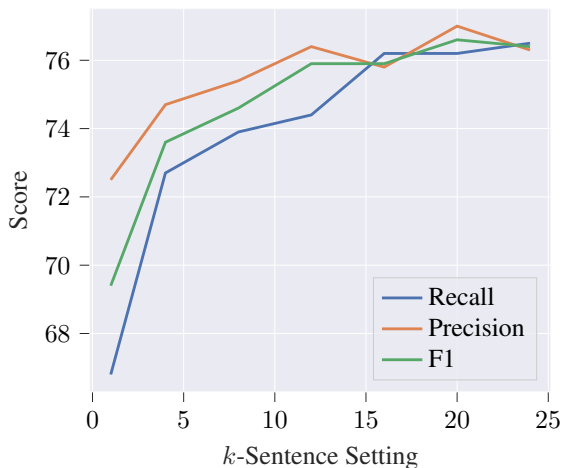

Figure 5: The CoNLL performance (average of *MUC*, B^3 and $CEAF_{\phi_4}$) of each k-Sentence-Incremental model on the OntoNotes dev set.

The results are shown in Figure 5. There are a few notable characteristics. The first is that as k increases, we see a much more dramatic lift when k is small (e.g. moving from 1 to 4 sentences) compared to when k is large. This effect corresponds to the intuition that coreferring expressions are usually close to their antecedent. The more coreference chains the model can observe simultaneously, the better it is at resolving them.

The second noteworthy trend is that increasing k improves recall (9.7%) far more than precision (3.8%). Although not shown here, we observe this trend across all three metrics within the CoNLL score (*MUC*, B^3 and $CEAF_{\phi_4}$). The result means that finding and resolving true coreference links (i.e. reducing false negatives) is a far more serious obstacle for the Sent-Inc model than for Part-Inc. Since the only difference in these models is how many embeddings are cached, the result suggests caching or not caching embeddings plays a large role in finding and correctly resolving mentions.

7 Future Work

A major goal would be to elevate incremental coreference resolvers to the same level as non-

incremental ones. As we showed in Section 6, a large part of the performance difference occurs because the XLNet encoder does not effectively handle incremental input. A simple strategy therefore may be to swap out that encoder for a more powerful one. However, few pre-trained language models targeted at NLU tasks are naturally incremental. One candidate is GPT-J (Wang and Komatsuzaki, 2021) but its size is prohibitively large.

Other ways to bridge this gap may come from improving the mention detection component. A similar task is nested named entity recognition, where the system must identify named entity boundaries and coarsely classify them. Recent nested NER systems such as Katiyar and Cardie (2018) or Yu et al. (2020a) may provide directions for improving mention detection in our incremental formulation.

8 Conclusion

We propose a sentence-incremental coreference resolution model using a shift-reduce formulation. The model delays mention clustering until the full span has been observed, alleviating a key flaw with previous incremental systems. It efficiently processes text, and avoids scoring a quadratic number of spans during mention detection.

In a sentence-incremental setting, our method outperforms strong baselines adapted from state-of-the-art systems. When access to the full document is allowed, the proposed system achieves similar performance to state-of-the-art methods while maintaining a higher level of incrementality. We investigate why this relaxation has such a dramatic effect, finding that the document encoder does not make effective use of its memory cache.

Our sentence-incremental results suggest an important point: non-incremental methods are not effective tools when they must be used incrementally. Creating new, incremental coreference resolvers that perform at the same level as non-incremental ones is a challenging but meaningful goal. Achieving this result would make a significant impact in downstream applications where text is received incrementally, such as dialogue systems or conversational question answering (e.g. Andreas et al. 2020; Martin et al. 2020). Our proposal demonstrates an important step towards highly effective, incremental coreference resolution systems.

Limitations

In this work, we have experimented with training neural networks on OntoNotes and evaluating on other datasets (the CODI-CRAC 2021 corpus). Several recently published papers have explored the difficulties of coreference resolution model transfer (Subramanian and Roth, 2019; Xia and Van Durme, 2021; Toshniwal et al., 2021; Yuan et al., 2022). These works have noted generalization problems with models trained on OntoNotes, with one particular difficulty that OntoNotes does not annotate singleton clusters, or 'markable' mentions.

Several recent works have addressed generalization issues by training on additional resources (Subramanian and Roth, 2019; Xia and Van Durme, 2021; Toshniwal et al., 2021; Yuan et al., 2022). In particular, Toshniwal et al. (2021) augment OntoNotes with *pseudo-singletons*: a fully trained coreference resolver scores all spans in the text, and the top-scoring spans outside of gold mentions are regarded as singletons. The authors show that adding pseudo-singletons to the OntoNotes training data improves (1) coreference resolution metrics on OntoNotes and (2) generalization capabilities. Pseudo-singletons are especially helpful for transfer learning from OntoNotes because other coreference datasets will often annotate singletons.

In our experiments, we attempted to use their published pseudo-singletons, but faced difficulties because the pseudo-singletons do not respect the "non-crossing bracketing" structure in OntoNotes, and overlap arbitrarily (not only nested). Our mention detector assumes mentions may be nested but otherwise do not overlap, and determining which pseudo-singletons to filter out without redoing the whole experiment was not feasible. We leave the problem for future work, but we agree that models trained on OntoNotes without heuristically added singletons are limited in their generalization capabilities.

Our experiments have focused on the *base* versions of XLNet and SpanBERT due to resource requirements. Training our models requires a GPU with 16 GB of memory; we used NVIDIA Tesla V100 16 GB cards. Greater memory efficiency could be achieved by extending the memory to be more dynamic. In the current system, entities are added but never evicted. Ideally, when a referent is no longer relevant to the context, it should be detected and removed. This concept has been explored with memory network-based systems (Liu et al., 2019; Toshniwal et al., 2020a), and also recent partly incremental systems (Xia et al., 2020; Toshniwal et al., 2020b). Memory-based systems using dynamic eviction strategies appear in other NLP tasks as well, such as semantic parsing (Jain and Lapata, 2021).

Ethical Considerations

NLP systems such as ours must be employed with special consideration that they do not demonstrate unwanted patterns towards protected groups. Previously, systems have been shown to learn harmful associations from training corpora. For example, Bolukbasi et al. (2016) show that word embeddings trained on a news corpus exhibit gender stereotypes, such as associating "receptionist" with "female".

Coreference resolution systems in particular may learn gender biases, and methods exist to counter this effect (Rudinger et al., 2018; Zhao et al., 2018). Our system is trained on OntoNotes, which include data from a diverse set of sources such as Wall Street Journal articles, telephone conversations and Bible passages. Our final trained model may therefore reflect undesirable content from these texts.

Any off-the-shelf deployment of our model should first check whether the model is harmful towards any protected group, and appropriate mitigation should be taken. For example, evaluating on specialized datasets such as Webster et al. (2018) may indicate whether the system unfairly predicts certain labels based on gender.

References

Emily Allaway, Shuai Wang, and Miguel Ballesteros. 2021. Sequential cross-document coreference resolution. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4659–4671, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Gerry Altmann and Mark Steedman. 1988. Interaction with context during human sentence processing. *Cognition*, 30(3):191–238.

Jacob Andreas, John Bufe, David Burkett, Charles Chen, Josh Clausman, Jean Crawford, Kate Crim, Jordan DeLoach, Leah Dorner, Jason Eisner, Hao Fang, Alan Guo, David Hall, Kristin Hayes, Kellie Hill, Diana Ho, Wendy Iwaszuk, Smriti Jha, Dan Klein, Jayant Krishnamurthy, Theo Lanman, Percy Liang, Christopher H. Lin, Ilya Lintsbakh, Andy McGovern, Aleksandr Nisnevich, Adam Pauls, Dmitrij Petters, Brent Read, Dan Roth, Subhro Roy, Jesse Rusak, Beth Short, Div Slomin, Ben Snyder, Stephon Striplin, Yu Su, Zachary Tellman, Sam Thomson, Andrei Vorobev, Izabela Witoszko, Jason Wolfe, Abby Wray, Yuchen Zhang, and Alexander Zotov. 2020. Task-oriented dialogue as dataflow synthesis. *Transactions of the Association for Computational Linguistics*, 8:556–571.

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *The first international conference on language resources and evaluation workshop on linguistics coreference*, volume 1, pages 563–566. Citeseer.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv:2004.05150*.

Tolga Bolukbasi, Kai-Wei Chang, James Zou, Venkatesh Saligrama, and Adam Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 4356–4364, Red Hook, NY, USA. Curran Associates Inc.

J. Carletta. 2006. Announcing the ami meeting corpus. In *The ELRA Newsletter 11(1)*, pages 3 – 5.

Haixia Chai and Michael Strube. 2022. Incorporating centering theory into neural coreference resolution. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2996–3002, Seattle, United States. Association for Computational Linguistics.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China. Association for Computational Linguistics.

John J. Godfrey, Edward C. Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Proceedings of the 1992 IEEE International Conference on Acoustics, Speech and Signal Processing - Volume 1*, ICASSP'92, page 517–520, USA. IEEE Computer Society.

Parag Jain and Mirella Lapata. 2021. Memory-Based Semantic Parsing. *Transactions of the Association for Computational Linguistics*, 9:1197–1212.

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.

Patrick Kahardipraja, Brielen Madureira, and David Schlangen. 2021. Towards incremental transformers: An empirical analysis of transformer models for incremental NLU. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1178–1189, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jungo Kasai, Hao Peng, Yizhe Zhang, Dani Yogatama, Gabriel Ilharco, Nikolaos Pappas, Yi Mao, Weizhu Chen, and Noah A. Smith. 2021. Finetuning pre-trained transformers into RNNs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10630–10643, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. 2020. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proceedings of the International Conference on Machine Learning (ICML)*.

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

Sopan Khosla, Juntao Yu, Ramesh Manuvinakurike, Vincent Ng, Massimo Poesio, Michael Strube, and Carolyn Rosé. 2021. The CODI-CRAC 2021 shared task on anaphora, bridging, and discourse deixis in dialogue. In *Proceedings of the CODI-CRAC 2021 Shared Task on Anaphora, Bridging, and Discourse Deixis in Dialogue*, pages 1–15, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Yuval Kirstain, Ori Ram, and Omer Levy. 2021. Coreference resolution without span representations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 14–19, Online. Association for Computational Linguistics.

Manfred Klenner and Don Tuggener. 2011. An incremental entity-mention model for coreference resolution with restrictive antecedent accessibility. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, pages 178–185, Hissar, Bulgaria. 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.

Fei Liu, Luke Zettlemoyer, and Jacob Eisenstein. 2019. The referential reader: A recurrent entity network for anaphora resolution. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5918–5925, Florence, Italy. Association for Computational Linguistics.

Tianyu Liu, Yuchen Eleanor Jiang, Nicholas Monath, Ryan Cotterell, and Mrinmaya Sachan. 2022. Transition-based structured prediction with autoregressive models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Abu Dhabi, United Arab Emirates and Virtual. Association for Computational Lingustics.

Robert L Logan IV, Andrew McCallum, Sameer Singh, and Dan Bikel. 2021. Benchmarking scalable methods for streaming cross document entity coreference. 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 4717–4731, Online. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 25–32, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Brielen Madureira and David Schlangen. 2020. Incremental processing in the age of non-incremental encoders: An empirical assessment of bidirectional models for incremental NLU. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 357–374, Online. Association for Computational Linguistics.

Scott Martin, Shivani Poddar, and Kartikeya Upasani. 2020. MuDoCo: Corpus for multidomain coreference resolution and referring expression generation. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 104–111, Marseille, France. European Language Resources Association.

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

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.

Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. Gender bias in

coreference resolution. 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 8–14, New Orleans, Louisiana. Association for Computational Linguistics.

Sanjay Subramanian and Dan Roth. 2019. Improving generalization in coreference resolution via adversarial training. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 192–197, Minneapolis, Minnesota. Association for Computational Linguistics.

Shubham Toshniwal, Allyson Ettinger, Kevin Gimpel, and Karen Livescu. 2020a. PeTra: A Sparsely Supervised Memory Model for People Tracking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5415–5428, Online. Association for Computational Linguistics.

Shubham Toshniwal, Sam Wiseman, Allyson Ettinger, Karen Livescu, and Kevin Gimpel. 2020b. 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.

Shubham Toshniwal, Patrick Xia, Sam Wiseman, Karen Livescu, and Kevin Gimpel. 2021. On generalization in coreference resolution. In *Proceedings of the Fourth Workshop on Computational Models of Reference, Anaphora and Coreference*, pages 111–120, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jack Urbanek, Angela Fan, Siddharth Karamcheti, Saachi Jain, Samuel Humeau, Emily Dinan, Tim Rocktäschel, Douwe Kiela, Arthur Szlam, and Jason Weston. 2019. Learning to speak and act in a fantasy text adventure game. 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 673–683, Hong Kong, China. Association for Computational Linguistics.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Sixth Message Understanding Conference (MUC-6): Proceedings of a Conference Held in Columbia, Maryland, November 6-8, 1995*.

Bailin Wang, Wei Lu, Yu Wang, and Hongxia Jin. 2018. A neural transition-based model for nested mention recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1011–1017, Brussels, Belgium. Association for Computational Linguistics.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. GitHub.

Xuewei Wang, Weiyan Shi, Richard Kim, Yoojung Oh, Sijia Yang, Jingwen Zhang, and Zhou Yu. 2019. Persuasion for good: Towards a personalized persuasive dialogue system for social good. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5635–5649, Florence, Italy. Association for Computational Linguistics.

Kellie Webster and James R. Curran. 2014. Limited memory incremental coreference resolution. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2129–2139, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Kellie Webster, Marta Recasens, Vera Axelrod, and Jason Baldridge. 2018. Mind the GAP: A balanced corpus of gendered ambiguous pronouns. *Transactions of the Association for Computational Linguistics*, 6:605–617.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston. 2013. Ontonotes release 5.0. *LDC2013T19, Philadelphia, Penn.: Linguistic Data Consortium*.

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.

Patrick Xia, João Sedoc, and Benjamin Van Durme. 2020. Incremental neural coreference resolution in constant memory. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8617–8624, Online. Association for Computational Linguistics.

Patrick Xia and Benjamin Van Durme. 2021. Moving on from OntoNotes: Coreference resolution model transfer. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5241–5256, Online and Punta Cana, Dominican Republic. 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.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019.

Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020a. Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6470–6476, Online. Association for Computational Linguistics.

Juntao Yu, Sopan Khosla, Nafise Moosavi, Silviu Paun, Sameer Pradhan, and Massimo Poesio. 2022. The universal anaphora scorer. In *Proceedings of the 13th Language Resources and Evaluation Conference*, Marseille, France. European Language Resources Association.

Juntao Yu, Alexandra Uma, and Massimo Poesio. 2020b. A cluster ranking model for full anaphora resolution. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 11–20, Marseille, France. European Language Resources Association.

Michelle Yuan, Patrick Xia, Chandler May, Benjamin Van Durme, and Jordan Boyd-Graber. 2022. Adapting coreference resolution models through active learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7533–7549, Dublin, Ireland. Association for Computational Linguistics.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. Gender bias in coreference resolution: Evaluation and debiasing methods. 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 15–20, New Orleans, Louisiana. Association for Computational Linguistics.

A   Action Constraints

To ensure the final state is always reached, it is necessary to enforce a set of rules during mention detection:

1. ADVANCE can only be called on the final token if the stack is empty.

2. POP and PEEK can only be called if the stack is non-empty.

3. PUSH can only be called once per token, ensuring that left boundaries are only marked once.

4. PUSH cannot directly follow POP or PEEK. Allowing this action sequence would either admit multiple paths to the same mention or non-nested overlapping mentions.

5. POP cannot directly follow PEEK, or else the same mention would be proposed twice.

6. PEEK cannot be called on the final token. This action would imply the stack is non-empty on the final token, and that POP must be called.

B k-Sentence-Incremental Mention Detection

We repeat the experiment in Section 6 for mention detection. For each k-Sentence-Incremental setting, we evaluate on the dev set and record the mention detection recall, precision and F1.

The results are shown in Figure 6. Certain trends remain the same as for the CoNLL score, namely that performance rises more when k is small compared to when it is large. However, we do not see the same dramatic difference in recall between $k = 1$ to $k = 24$ settings as in Section 6. Here, the difference in recall between the two settings is around 3%, whereas in Figure 5 it is 9.7%.

Overall, the reduced severity between $k = 1$ and $k = 24$ settings compared to Figure 5 most likely indicate that XLNet's caching deficiencies affect mention clustering (particularly false negatives) more seriously than mention detection.
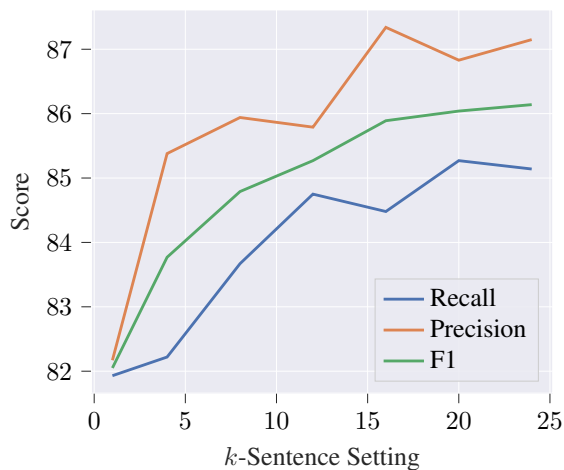

Figure 6: Mention Detection performance (average of *MUC*, B^3 and $CEAF_{\phi_4}$) of each k-Sentence-Incremental model on the OntoNotes dev set.

C   Partitioning Document Clusters

We explore the deficiency in the previous section further, guided by the hypothesis that XLNet relies on active inputs and cannot effectively use its memory. We partition each document into segments of k sentences, and call the number of sentences in
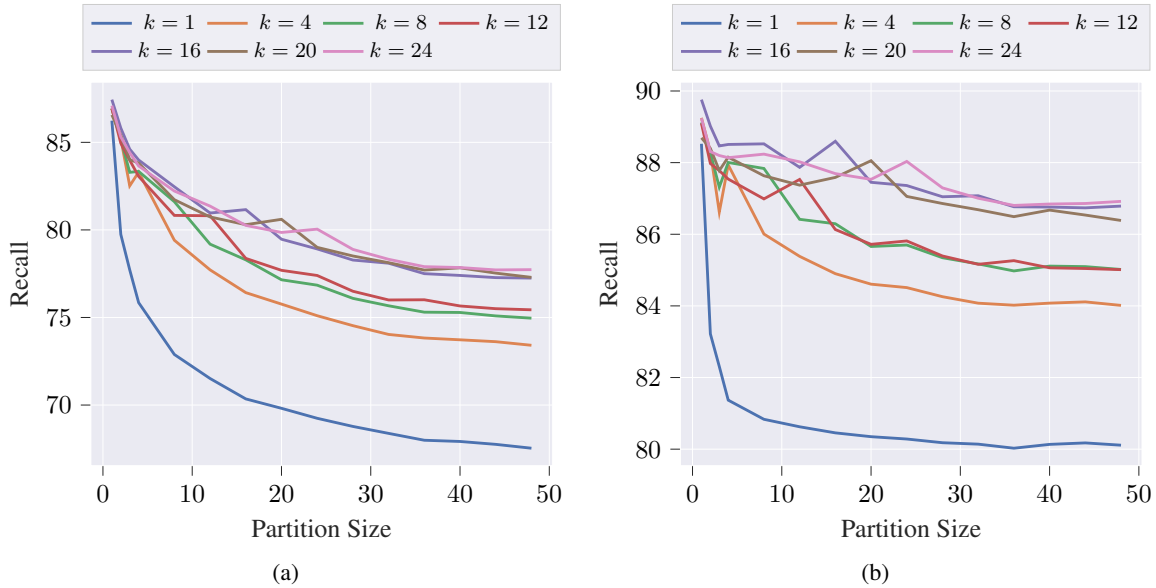
Figure 7: Evaluation results on the OntoNotes dev set when the gold labels and k-Sentence-Incremental predictions are partitioned according to various sizes. Figure 7a shows the CoNLL recall scores for coreference resolution. Figure 7b shows the mention detection recall scores. Notice that whenever k is equal to the partition size, there is a noticeable performance increase, indicating that XLNet relies heavily on active inputs rather than its memory.

each segment the 'partition size'. Within each segment, we maintain the original coreference links. However, we remove the coreference links between segments. For each k-Sentence-Incremental model, we similarly partition their coreference predictions, and evaluate against the partitioned gold labels on the OntoNotes dev set.

Each segment therefore is independent from the other, and we can measure how reliant the model is on its active inputs by observing performance change across partition sizes. For example, when the partition size is 1, the only coreference links are intra-sentential ones. In this case, models are only evaluated on their intra-sentential coreference resolution ability. When the partition size is large, the models are evaluated on the documents' original coreference chains. Since the previous experiments demonstrated shifts in incrementality heavily affected recall, we measure the mention detection recall and CoNLL recall score.

The results are shown in Figure 7. All models do well when the partition size is one, reflecting the fact that intra-sentential coreference is generally simpler than distantly linked mentions. As the partition size increases, model performance decreases as the coreference chains become more spread apart and raise the task difficulty. Crucially, we notice a upward performance bump whenever the partition size matches the k-Sentence-Incremental set-

ting, for both coreference performance and mention detection When k matches the partition size, the model observes coreference chains that are always within the active input window. This performance bump therefore indicates XLNet is much better at mention detection and coreference when the coreference chain occurs within its active inputs. Performance suffers whenever the model must rely more on its memory (whenever k is not equal to the partition size). In particular, these results suggest that more powerful pre-trained language models, in particular ones that can take better advantage of cached representations, may be more successful at incremental coreference resolution.

D Speaker Embeddings

The ICoref-*inc* model from Xia et al. (2020) is an important comparison point as the only baseline in the sentence-incremental setting. While ICoref-*inc* does not rely on speaker embeddings, our own models (both Part-Inc and Sent-Inc) do. Given the important role of speaker identity in a dialogue setting, it is useful to know the effect of removing these embeddings in our models.

We compare the Sent-Inc model with and without speaker embeddings in Table 4 for OntoNotes, and Table 5 for CODI-CRAC. We find that speaker embeddings play little to no role in coreference performance. In OntoNotes, removing speaker

Model	MUC			B^3			$CEAF_{\phi_4}$			Avg.
	Rec.	Prec.	F1	Rec.	Prec.	F1	Rec.	Prec.	F1	F1
Sent-Inc	76.4	78.7	77.6	68.0	68.2	68.1	56.1	70.6	62.6	69.4
−speaker	76.4	79.0	77.7	68.0	68.6	68.3	56.3	70.8	62.7	69.5

Table 4: Results on the OntoNotes dev set comparing the Sent-Inc model with and without speaker embeddings.

Model	LIGHT	AMI	PERS.	SWBD.	Avg.
Sent-Inc	50.6	32.5	48.5	44.4	44.0
−speaker	50.1	32.0	49.6	43.3	43.8

Table 5: Results on the CODI-CRAC dev set comparing the Sent-Inc model with and without speaker embeddings.

embeddings improves CoNLL F1 by 0.1, and in CODI-CRAC, it decreases performance by 0.2 F1. In both cases, the results are unlikely to be statistically significant. The finding indicates that Sent-Inc's advantage over ICoref-*inc* is not simply due to feature selection but a true modelling advantage. It also suggests that further performance improvements are possible if speaker identity can be better represented, since Sent-Inc effectively ignores the speaker embeddings. One possibility, from Wu et al. (2020), is to preprocess the text with speaker tags directly included in the input, rather than including it separately. This way, the document encoder directly learns how to handle speakers, instead of relying on a separate embedding in downstream classifiers.

E XLNet in Non-Incremental Baselines

Choosing XLNet as the document encoder is motivated by the fact that XLNet can efficiently cache and reuse input, making it suitable for incremental processing. However, XLNet can also be used non-incrementally in the same way as SpanBERT. In particular, we can train Joshi et al. (2020)'s coreference system using XLNet instead of Span-BERT. This experiment allows us to compare how the choice of pre-trained language model affects performance.

Table 6 shows results of training Joshi et al. (2020) with an XLNet encoder instead of Span-BERT on the OntoNotes dev set. XLNet significantly underperforms compared to SpanBERT, scoring almost 7 CoNLL F1 points lower. Surprisingly, XLNet is an effective document encoder for our Part-Incremental formulation (achieving 76.3 F1 on the OntoNotes test set), but ineffective when

used in Joshi et al. (2020)'s non-incremental setup. We do not attempt swapping the fine-tuned XLNet into Toshniwal et al. (2020b) or Xia et al. (2020) as it seems unlikely to yield useful results.

F Hyperparameters and Other Model Details

The main hyperparameters are listed in Table 7.

The bottom four rows refer to the maximum number of learned embeddings we use for each feature. Additionally:

- The top performing Part-Inc model uses 20 sentences as active input, with the remainder as memory (up to 512 tokens total).

- During training, the Sent-Inc model accumulates gradients after every 32 sentences to ensure that the memory used does not exceed capacity.

Our implementation is based off of Xu and Choi (2020)'s codebase. We find their model hyperparameters are already extremely well-tuned, and so we do not explore further hyperparameter tuning for these cases. Regarding new hyperparameters introduced in this work, we follow previous work in choosing sensible values. For example, the StackLSTM and Action History LSTM hidden sizes follow Dyer et al. (2015)'s recommendations.

We train all models using NVIDIA Tesla V100 16 GB cards on an HPC cluster. Training convergence takes approximately 24 hours. Both Sent-Inc and Part-Inc models contain around 140 million parameters.

G Dataset Details

For all datasets, we follow standard preprocessing steps such as tokenization, mapping subword units to token IDs, and adding segment boundary tokens (such as [CLS] and [SEP]). Since our algorithms rely on teacher forcing, we compute gold actions for both mention detection and mention clustering steps.

The English portion of OntoNotes 5.0 contains 3493 documents, divided into 2802/343/348

Model	MUC			B^3			$CEAF_{\phi_4}$			Avg.
	Rec.	Prec.	F1	Rec.	Prec.	F1	Rec.	Prec.	F1	F1
SpanBERT+*e2e-coref*	82.5	84.3	83.4	75.8	76.8	76.3	71.7	74.7	73.1	77.6
XLNet+*e2e-coref*	70.3	80.8	75.2	63.7	73.0	68.0	67.5	70.4	68.9	70.7

Table 6: Results from non-incremental methods on the OntoNotes dev set with different pretrained language models. *e2e-coref* refers to the non-incremental formulation from Lee et al. (2018), which is adapted in Joshi et al. (2020).

Hyperparameter	Value
Encoder Learning Rate	2e-5
Task Learning Rate	1e-4
Adam Eps	1e-6
Adam Weight Decay	1e-2
Gradient Clipping Norm	1
Dropout Rate	0.3
StackLSTM Hidden Size	200
Action History Hidden Size	30
f_M Hidden Size	1000
f_C Hidden Size	3000
New Cell Threshold (α)	0.0
ϕ_C Max Entity Count	10
ϕ_C Max Mention Distance	10
ϕ_M Max Span Width	30
Max Speaker Number	20

Table 7: Hyperparameters used during training.

H Evaluation Metrics

As mentioned in Section 4, we evaluate using the MUC, B^3 and $CEAF_{\phi_4}$ metrics, together with their average – commonly referred to as the CoNLL score after its usage in the CoNLL 2012 Shared Task (Pradhan et al., 2012).

None of the metrics have a trivial definition. Each metric measures different aspects of the predicted coreference chains. We do not provide a full definition of each metric here, but note that MUC corresponds to a *link*-based measure, B^3 is a *mention*-based measure, and $CEAF_{\phi_4}$ is an *entity*-based measure. We refer the reader to the CoNLL 2012 Shared Task paper (Pradhan et al., 2012) for an overview of all three metrics and to the original papers for a full description (Vilain et al., 1995; Bagga and Baldwin, 1998; Luo, 2005).

splits for training, validation and test sections, respectively. The corpus contains examples for English, Chinese and Arabic; however, we only use the English portion in this paper. Details to download the dataset can be found at: https://catalog.ldc.upenn.edu/LDC2013T19. The official scorer is available at https://github.com/conll/reference-coreference-scorers. We do not remove data from OntoNotes.

The CODI-CRAC 2021 corpus contains 134 English documents, split into 60 documents for validation and 74 documents for testing. Since we use the corpus solely for evaluation, we only use the test set. The corpus is available at: https://codalab.lisn.upsaclay.fr/competitions/614, although a license from the Linguistic Data Corsortium (LDC) is needed for the Switchboard portion. The official scorer can be found at: https://github.com/juntaoy/universal-anaphora-scorer. As detailed in Section 4.1, we remove singletons due to lack of training data.