Learning to Ignore: Long Document Coreference with Bounded Memory Neural Networks

Shubham Toshniwal[1], Sam Wiseman[1], Allyson Ettinger[2], Karen Livescu[1], Kevin Gimpel[1]
[1]Toyota Technological Institute at Chicago
[2]Department of Linguistics, University of Chicago
{shtoshni, swiseman, klivescu, kgimpel}@ttic.edu, aettinger@uchicago.edu

Abstract

Long document coreference resolution remains a challenging task due to the large memory and runtime requirements of current models. Recent work doing incremental coreference resolution using just the global representation of entities shows practical benefits but requires keeping all entities in memory, which can be impractical for long documents. We argue that keeping all entities in memory is unnecessary, and we propose a memory-augmented neural network that tracks only a small bounded number of entities at a time, thus guaranteeing a linear runtime in length of document. We show that (a) the model remains competitive with models with high memory and computational requirements on OntoNotes and LitBank, and (b) the model learns an efficient memory management strategy easily outperforming a rule-based strategy.

1 Introduction

Long document coreference resolution poses runtime and memory challenges. Current best models for coreference resolution have large memory requirements and quadratic runtime in the document length (Joshi et al., 2019; Wu et al., 2020), making them impractical for long documents.

Recent work revisiting the entity-mention paradigm (Luo et al., 2004; Webster and Curran, 2014), which seeks to maintain explicit representations only of entities, rather than all their constituent mentions, has shown practical benefits for memory while being competitive with state-of-the-art models (Xia et al., 2020). In particular, unlike other approaches to coreference resolution which maintain representations of both mentions *and* their corresponding entity clusters (Rahman and Ng, 2011; Stoyanov and Eisner, 2012; Clark and Manning, 2015; Wiseman et al., 2016; Lee et al., 2018) , the entity-mention paradigm stores representations only of the entity clusters, which are updated incrementally as coreference predictions are made. While such an approach requires less memory than those that additionally store mention representations, the number of entities can still become impractically large when processing long documents, making the storing of all entity representations problematic.

Is it necessary to maintain an unbounded number of mentions or entities? Psycholinguistic evidence suggests it is not, as human language processing is incremental (Tanenhaus et al., 1995; Keller, 2010) and has limited working memory (Baddeley, 1986). In practice, we find that most entities have a small spread (number of tokens from first to last mention of an entity), and thus do not need to be kept persistently in memory. This observation suggests that tracking a limited, small number of entities at any time can resolve the computational issues, albeit at a potential accuracy tradeoff.

Previous work on finite memory models for coreference resolution has shown potential, but has been tested only on short documents (Liu et al., 2019; Toshniwal et al., 2020). Moreover, this previous work makes token-level predictions while standard coreference datasets have span-level annotations. We propose a finite memory model that performs quasi-online coreference resolution,[1] and test it on LitBank (Bamman et al., 2020) and OntoNotes (Pradhan et al., 2012). The model is trained to manage its limited memory by predicting whether to "forget" an entity already being tracked in exchange for a new (currently untracked) entity. Our empirical results show that: (a) the model is competitive with an unbounded memory version, and (b) the model's learned memory management outperforms a strong rule-based baseline.[2]

[1]"Quasi-online" because document encoding uses bidirectional transformers with access to future tokens.

[2]Code at https://github.com/shtoshni92/

Table 1: Max. Total Entity Count vs. Max. Active Entity Count.

	LitBank	OntoNotes
Max. Total Entity Count	199	94
Max. Active Entity Count	18	24

2 Entity Spread and Active Entities

Given input document $\mathcal{D}$, let $(x_n)_{n=1}^{N}$ represent the N mention spans corresponding to M underlying entities $(e_m)_{m=1}^{M}$. Let START(x_i) and END(x_i) denote the start and end token indices of the mention span x_i in document $\mathcal{D}$. Let ENT(x_i) denote the entity of which x_i is a mention. Given this notation we next define the following concepts.

Entity Spread Entity spread denotes the interval of token indices from the first mention to the last mention of an entity. The entity spread ES(e) of entity e is given by:

$$\text{ES}(e) = [\min_{\text{ENT}(x)=e} \text{START}(x), \max_{\text{ENT}(x)=e} \text{END}(x)]$$

Active Entity Count Active entity count AE(t) at token index t denotes the number of unique entities whose spread covers the token t, i.e., $\text{AE}(t) = |\{e \mid t \in \text{ES}(e)\}|$.

Maximum Active Entity Count Maximum active entity count MAE$(\mathcal{D})$ for a document $\mathcal{D}$ denotes the maximum number of active entities at any token index in $\mathcal{D}$, i.e., $\text{MAE}(\mathcal{D}) = \max_{t\in[|\mathcal{D}|]} \text{AE}(t)$. This measure can be simply extended to a corpus $\mathcal{C}$ as: $\text{MAE}(\mathcal{C}) = \max_{\mathcal{D}\in\mathcal{C}} \text{MAE}(\mathcal{D})$.

Table 1 shows the MAE and the maximum total entity count in a single document, for LitBank and OntoNotes. For both datasets the maximum active entity count is much smaller than the maximum total entity count. Thus, rather than keeping all the entities in memory at all times, models can in principle simply focus on the far fewer active entities at any given time.

3 Model

Based on the preceding finding, we will next describe models that require tracking only a small, bounded number of entities at any time.

To make coreference predictions for a document, we first encode the document and propose candidate mentions. The proposed mentions are then processed sequentially and are either: (a) added to an existing entity cluster, (b) added to a new cluster, (c) ignored due to limited memory capacity (for bounded memory models), or (d) ignored as an invalid mention.

Document Encoding is done using the SpanBERT$_{\text{LARGE}}$ model finetuned for OntoNotes and released as part of the coreference model of Joshi et al. (2020). We don't further finetune the SpanBERT model. To encode long documents, we segment the document using the *independent* and *overlap* strategies described in Joshi et al. (2019).[3] In *overlap* segmentation, for a token present in overlapping BERT windows, the token's representation is taken from the BERT window with the most neighboring tokens of the concerned token. For both datasets we find that *overlap* slightly outperforms *independent*.

Mention Proposal Given the encoded document, we next predict the top-scoring mentions which are to be clustered. The goal of this step is to have high recall, and we follow previous work to threshold the number of spans chosen (Lee et al., 2017). Given a document $\mathcal{D}$, we choose $0.3\times|\mathcal{D}|$ top spans for LitBank, and $0.4 \times |\mathcal{D}|$ for OntoNotes.

Note that we pretrain the mention proposal model before training the mention proposal and mention clustering pipeline end-to-end, as done by Wu et al. (2020). The reason is that without pretraining, most of the mentions proposed by the mention proposal model would be invalid mentions, i.e., spans that are not mentions, which would not provide any training signal to the mention clustering stage. For both datasets, we sample invalid spans with 0.2 probability during training, so as to roughly equalize the number of invalid spans and actual mentions, as suggested by Xia et al. (2020).

Mention Clustering Let $(x_i)_{i=1}^{K}$ represent the top-K candidate mention spans from the mention proposal step and let $s_m(x_i)$ represent the mention score for span x_i, which indicates how likely it is that a span constitutes a mention. Assume that the mentions are already ordered based on their position in the document and are processed sequentially in that order.[4] Let $E = (e_m)_{m=1}^{M}$ represent the M

long-doc-coref

[3]We modify the *overlap* segmentation to respect sentence boundary or token boundary when possible.

[4]Specifically, they are ordered based on START$(\cdot)$ index with ties broken using END$(\cdot)$.

entities currently being tracked by the model (initially $M = 0$). For ease of discussion, we will overload the terms x_i and e_j to also correspond to their respective representations.

In the *first* step, the model decides whether the span x_i refers to any of the entities in E as follows:

$$s_c(x_i, e_j) = f_c([x_i; e_j; x_i \odot e_j; g(x_i, e_j)]) + s_m(x_i)$$
$$s_c^{top} = \max_{j=1\ldots M} s_c(x_i, e_j)$$
$$e^{top} = \arg\max_{j=1\ldots M} s_c(x_i, e_j)$$

where $\odot$ represents the element-wise product, and $f_c(\cdot)$ corresponds to a learned feedforward neural network. The term $g(x_i, e_j)$ correponds to a concatenation of feature embeddings that includes embeddings for (a) number of mentions in e_j, (b) number of mentions between x_i and last mention of e_j, (c) last mention decision, and (d) document genre (only for OntoNotes).

Now if $s_c^{top} > 0$ then x_i is considered to refer to e^{top}, and e^{top} is updated accordingly.[5] Otherwise, x_i does not refer to any entity in E and a *second* step is executed, which will depend on the choice of memory architecture. We test three memory architectures, described below.

1. **Unbounded Memory (U-MEM)**: If $s_m(x_i) > 0$ then we create a new entity $e_{M+1} = x_i$ and append it to E. Otherwise the mention is ignored as invalid, i.e., it doesn't correspond to an entity. This differs from Xia et al. (2020) who append all non-coreferent mentions. The reason for the change is that appending all mentions can hurt performance on LitBank where singletons are explicitly marked and used for evaluation.

2. **Bounded Memory**: Suppose the model has a capacity of tracking C entities at a time. If $C > M$, i.e., the memory capacity has not been fully utilized, then the model behaves like U-MEM. Otherwise, the bounded memory models must decide between: (a) evicting an entity already being tracked, (b) ignoring x_i due to limited capacity, and (c) ignoring the mention as invalid. We test two bounded memory variants that are described below.

(a) **Learned Bounded Memory (LB-MEM)**: The proposed LB-MEM architecture tries to predict a score $f_r(.)$ corresponding to the anticipated number of remaining mentions for any entity or mention, and compares it against the mention score $s_m(x_i)$ as follows:

$$d = \arg\min[f_r(e_1), \ldots, f_r(e_M), f_r(x_i), s_m(x_i)]$$

where $f_r(\cdot)$ is a learned feedforward neural network. If $1 \leq d \leq M$ then then the model evicts the previous entity e_d and reinitialize it to x_i. Otherwise if $d = M + 1$ then the model ignores x_i due to limited capacity. Finally if $d = M + 2$ then the model predicts the mention to be invalid.

(b) **Rule-based Bounded Memory (RB-MEM)** The Least Recently Used (LRU) principle is a popular choice among memory models (Rae et al., 2016; Santoro et al., 2016). While LB-MEM considers all potential entities for eviction, with RB-MEM this choice is restricted to just the LRU entity, i.e., the entity whose mention was least recently seen. The rest of the steps are similar to the LB-MEM model.

Training All the models are trained using teacher forcing. The ground truth decisions for bounded memory models are chosen to maximize the number of mentions tracked by the model (details in Appendix A.3). Finally, the training loss is calculated via the addition of the cross-entropy losses for the two steps of mention clustering.

Table 2: Results for LitBank (CoNLL F1).

Model	Dev F1	Test F1
U-MEM	76.5	75.9
LB-MEM		
5 cells	70.6	69.5
10 cells	75.4	74.9
20 cells	76.3	75.7
RB-MEM		
5 cells	67.5	66.7
10 cells	72.2	71.8
20 cells	73.1	72.6
Bamman et al. (2020)	-	68.1

4 Experimental Setup

4.1 Datasets

LitBank is a recent coreference dataset for literary texts (Bamman et al., 2020). The dataset consists of prefixes of 100 novels with an average length of 2100 words. Singletons are marked and used for evaluation. Evaluation is done via 10-fold cross-validation over 80/10/10 splits.[6]

[5]We use weighted averaging where the weight for e^{top} corresponds to the number of previous mentions seen for e^{top}.

[6]https://github.com/dbamman/lrec2020-coref/tree/master/data

Table 3: Results for OntoNotes (CoNLL F1) .

Model	Dev F1	Test F1
U-MEM	77.7	77.4
LB-MEM		
5 cells	73.1	73.0
10 cells	76.6	76.2
20 cells	77.7	77.3
RB-MEM		
5 cells	69.0	68.8
10 cells	75.2	75.0
20 cells	77.5	77.5
U-MEM (Xia et al., 2020)	78.7	78.2
Joshi et al. (2020)	80.1	79.6
Wu et al. (2020)	83.4	83.1

OntoNotes consists of 2802/343/348 documents in the train/development/test splits, respectively (Pradhan et al., 2012). The documents span 7 genres and have an average length of 463 words. Singletons are not marked in the dataset.

4.2 Hyperparameters

Document encoding is done using the SpanBERT$_{\text{LARGE}}$ model of Joshi et al. (2020) which was finetuned for OntoNotes. The SpanBERT model is not further finetuned. The other model parameters are trained using the Adam optimizer (Kingma and Ba, 2014) with an initial learning rate of 5×10^{-4}. For span representation, we use the embedding function described in Lee et al. (2017). For OntoNotes we follow the setup of Xia et al. (2020). We differ, however, in training all the model parameters, except SpanBERT, from scratch. The models are trained for 10 epochs with a patience of 3 epochs, i.e., reduce learning rate by a 0.1 factor if the validation loss doesn't improve for 3 epochs. For LitBank the models are trained for 25 epochs with a patience of 3 epochs. For more details see Appendix A.2.

5 Results

Tables 2 and 3 show results of all the proposed models for LitBank and OntoNotes respectively. As expected, the bounded memory models improve with increase in memory. For both datasets, the LB-MEM model with 20 memory cells is competitive with the U-MEM model. The RB-MEM model with 20 memory cells is competitive on OntoNotes but is significantly worse than the other two on LitBank. Comparing among the bounded memory models, the LB-MEM model is significantly better than RB-MEM for lower numbers of memory cells.

Table 4: Peak memory and inference time statistics for the LitBank cross-validation split zero.

Model	Peak training mem. (in GB)	Peak inference mem. (in GB)	Inference time (in s)
U-MEM	11.6	3.1	29.25
LB-MEM			
5 cells	8.0	3.2	27.31
10 cells	8.4	3.2	27.44
20 cells	9.1	3.2	27.86
RB-MEM			
5 cells	8.0	3.2	26.19
10 cells	8.3	3.2	26.50
20 cells	8.9	3.2	26.19

Table 5: Comparison of number of entities in memory.

Model	LitBank		OntoNotes	
	Avg	Max	Avg	Max
U-MEM	97.0	198	16.3	87
LB-MEM				
5 cells	5.0	5	4.6	5
10 cells	10.0	10	8.1	10
20 cells	20.0	20	12.4	20
RB-MEM				
5 cells	5.0	5	4.6	5
10 cells	10.0	10	8.1	10
20 cells	20.0	20	12.4	20

We analyze the reasons for this in the next section.

Between the two datasets, we see that the increase in memory results in larger improvement for LitBank. We also establish a new state-of-the-art for LitBank with the U-MEM memory model. For OntoNotes, our models are competitive with comparable models such as Xia et al. (2020). The performance difference between the two U-MEM models might be because we try to predict invalid mentions which, while beneficial for LitBank, can lead to lower mention recall for OntoNotes. We expect gains by further finetuning the SpanBERT model and learning a parameterized global entity representation, but we leave them for future work.

6 Analysis

In this section we analyze the behavior of the three memory models on LitBank and OntoNotes.

Memory Utilization Table 4 compares the memory and inference time statistics for the different memory models for the LitBank cross-validation split zero.[7] For training, the bounded memory models are significantly less memory intensive than the U-MEM model. The table also shows that the

[7]Peak memory usage estimated via `torch.cuda.max_memory_allocated()`

Table 6: Average number of mentions ignored by the two bounded memory models.

Memory size	LitBank LB-MEM	LitBank RB-MEM	OntoNotes LB-MEM	OntoNotes RB-MEM
5	18.3	83.2	0.5	5.4
10	0.0	34.5	0.0	0.7
20	0.0	7.0	0.0	0.0

bounded memory models are faster than the U-MEM memory model during inference (inference time calculated by averaging over three runs). This is because the number of entities tracked by the U-MEM memory model grows well beyond the maximum of 20 memory slots reserved for the bounded models as shown in Table 5.

Surprisingly, for inference we see that the bounded models have a slightly larger memory footprint than the U-MEM model. This is because the document encoder, SpanBERT, dominates the memory usage during inference (as also observed by Xia et al., 2020). Thus the peak memory usage during inference is determined by the mention proposal stage rather than the mention clustering stage. And during the mention proposal stage, the additional parameters of bounded memory models, which are loaded as part of the whole model, cause the slight uptick in peak inference memory. Note that using a cheaper encoder or running on a sufficiently long document, such as a book, can change these results.

Number of Entities in Memory Table 5 compares the maximum number of entities kept in memory by the different memory models for the LitBank cross-validation dev sets and the OntoNotes dev set. As expected, the U-MEM model keeps more entities in memory than the bounded memory models on average for both datasets. For LitBank the difference is especially stark with the U-MEM model tracking about 5/10 times more entities in memory on average/worst case, respectively. Also, while some OntoNotes documents do not use even the full 5 memory cell capacity, all LitBank documents fully utilize even the 20 memory cell capacity. This is because LitBank documents are more than four times as long as OntoNotes documents, and LitBank has singletons marked. These results also justify our initial motivation that with long documents, the memory requirement will increase even if we only keep the entity representations.

Table 7: Error Analysis for OntoNotes dev set. CE=Conflated Entities, DE=Divided Entity, EM=Extra Mention, EE=Extra Entity, MM=Missing Mention, ME=Missing Entity.

Model	CE	DE	EM	EE	MM	ME
U-MEM	950	901	635	621	493	542
LB-MEM						
5 cells	722	1020	394	426	982	1058
10 cells	863	988	499	505	637	719
20 cells	894	905	571	542	513	631
RB-MEM						
5 cells	724	1166	386	406	989	1335
10 cells	851	1088	474	547	702	749
20 cells	880	903	559	561	531	634

LB-MEM vs. RB-MEM Table 6 compares the number of mentions ignored by LB-MEM and RB-MEM. The LB-MEM model ignores far fewer mentions than RB-MEM. This is because while the RB-MEM model can only evict the LRU entity, which might not be optimal, the LB-MEM model can choose any entity for eviction. These statistics combined with the fact that the LB-MEM model typically outperforms RB-MEM mean that the LB-MEM model is able to anticipate which entities are important and which are not.

Error Analysis Table 7 presents the results of automated error analysis done using the Berkeley Coreference Analyzer (Kummerfeld and Klein, 2013) for the OntoNotes dev set. As the memory capacity of models increases, the errors shift from missing mention, missing entity, and divided entity categories, to conflated entities, extra mention, and extra entity categories. For the 5-cell configuration, the LB-MEM model outperforms RB-MEM in terms of tracking more entities.

7 Conclusion and Future Work

We propose a memory model which tracks a small, bounded number of entities. The proposed model guarantees a linear runtime in document length, and in practice significantly reduces peak memory usage during training. Empirical results on LitBank and OntoNotes show that the model is competitive with an unbounded memory version and outperforms a strong rule-based baseline. In particular, we report state of the art results on LitBank. In future work we plan to apply our model to longer, book length documents, and plan to add more structure to the memory.

Acknowledgments

We thank David Bamman for help with the LitBank setup, and Patrick Xia for answering questions about their coreference model. We also thank the anonymous ACL reviewers for their valuable feedback. This material is based upon work supported by the National Science Foundation under Award Nos. 1941178 and 1941160.

References

Alan Baddeley. 1986. *Working Memory*. Oxford University Press.

David Bamman, Olivia Lewke, and Anya Mansoor. 2020. An Annotated Dataset of Coreference in English Literature. In *LREC*.

Kevin Clark and Christopher D. Manning. 2015. Entity-Centric Coreference Resolution with Model Stacking. In *ACL*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving Pre-training by Representing and Predicting Spans. *TACL*, 8.

Mandar Joshi, Omer Levy, Luke Zettlemoyer, and Daniel Weld. 2019. BERT for Coreference Resolution: Baselines and Analysis. In *EMNLP*.

Frank Keller. 2010. Cognitively Plausible Models of Human Language Processing. In *ACL*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *ICLR*.

Jonathan K. Kummerfeld and Dan Klein. 2013. Error-Driven Analysis of Challenges in Coreference Resolution. In *EMNLP*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end Neural Coreference Resolution. In *EMNLP*.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-Order Coreference Resolution with Coarse-to-Fine Inference. In *NAACL-HLT*.

Fei Liu, Luke Zettlemoyer, and Jacob Eisenstein. 2019. The Referential Reader: A Recurrent Entity Network for Anaphora Resolution. In *ACL*.

Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. 2004. A Mention-Synchronous Coreference Resolution Algorithm Based On the Bell Tree. In *ACL*.

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*, CoNLL '12.

Jack W. Rae, Jonathan J. Hunt, Ivo Danihelka, Timothy Harley, Andrew W. Senior, Gregory Wayne, Alex Graves, and Tim Lillicrap. 2016. Scaling Memory-Augmented Neural Networks with Sparse Reads and Writes. In *NeurIPS*.

Altaf Rahman and Vincent Ng. 2011. Narrowing the modeling gap: a cluster-ranking approach to coreference resolution. *Journal of Artificial Intelligence Research*, 40:469–521.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy P. Lillicrap. 2016. One-shot Learning with Memory-Augmented Neural Networks. In *ICML*.

Veselin Stoyanov and Jason Eisner. 2012. Easy-first Coreference Resolution. In *COLING*.

MK Tanenhaus, MJ Spivey-Knowlton, KM Eberhard, and JC Sedivy. 1995. Integration of visual and linguistic information in spoken language comprehension. *Science*, 268(5217).

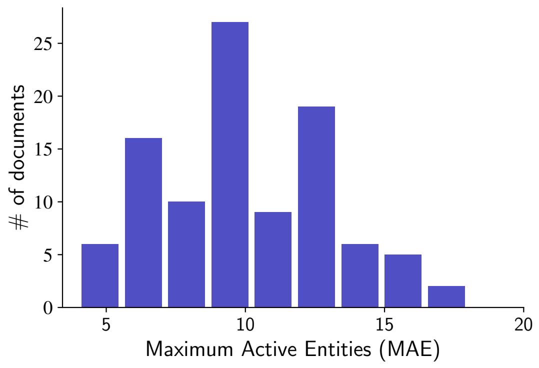
Shubham Toshniwal, Allyson Ettinger, Kevin Gimpel, and Karen Livescu. 2020. PeTra: A Sparsely Supervised Memory Model for People Tracking. In *ACL*.

Kellie Webster and James R. Curran. 2014. Limited memory incremental coreference resolution. In *COLING*.

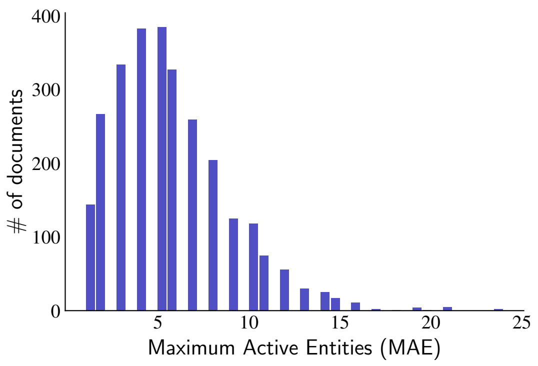
Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. 2016. Learning Global Features for Coreference Resolution. In *NAACL*.

Wei Wu, Fei Wang, Arianna Yuan, Fei Wu, and Jiwei Li. 2020. Coreference Resolution as Query-based Span Prediction. In *ACL*.

Patrick Xia, João Sedoc, and Benjamin Van Durme. 2020. Revisiting Memory-Efficient Incremental Coreference Resolution. In *EMNLP*.

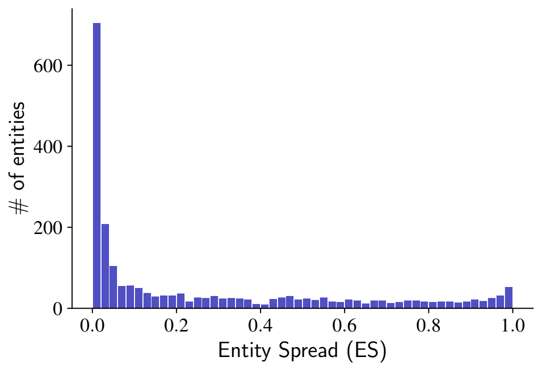


(a) LitBank.

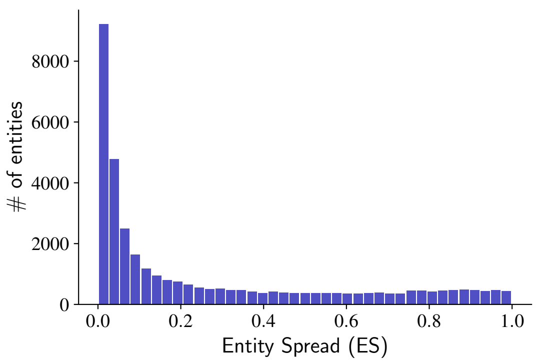


(b) OntoNotes Training Set.

Figure 1: Histograms of Maximum Active Entities for documents in LitBank and OntoNotes.



(a) LitBank.



(b) OntoNotes Training Set.

Figure 2: Histograms of Entity Spread as fraction of document length for LitBank and OntoNotes.

A Appendix

A.1 Maximum Active Entities

Figure 1 visualizes the histograms of length of Entity Spread (ES), defined in Section 2, as a fraction of document length for documents in LitBank and OntoNotes. For LitBank we only visualize the entity spread of non-singleton clusters because otherwise the histogram is too skewed towards one. Figure 2 visualizes the histograms of Maximum Active Entity Count (MAE), defined in Section 2, for documents in LitBank and OntoNotes.

A.2 Model Details

Other hyperparameters We stick with the hyperparameters for feedforward neural network (FFNN) size and depth, and dropout from Joshi et al. (2020). One hyperparameter that we find to be important is the weight of the non-coreferent term in the cross-entropy loss for the first step of mention clustering. We find that placing a higher weight of 2.0 on that term leads to consistent performance gains. This might be because of that term's significance, as the value of that term decides whether the next step of mention clustering is triggered or not.

Table 8: Hyperparameter options with the bold choices highlighted as bold.

Parameter	Range
Dropout	{0.3}
FFNN hidden layer	{3000}
FFNN # of hidden layers	1
Document Encoding	{Independent, **Overlap**}
Non-coreferent entity weight	{1.0, **2.0**, 5.0}

Expected Validation Performance Since LitBank has 10 cross-validation splits, the grid search based tuning process was limited to a few cross-validation splits. For LitBank, in our initial experiments with gold mention clustering we find that *overlap* segmentation gave a gain of about 0.5% F1 and we stuck with the choice from then onwards. For non-coreferent entity weight, we see an improvement of 0.5-1% F1 on going from 1.0 to 2.0 but the performance with 5.0 weight drops below of that with 1.0.

For OntoNotes, we find that deviating from *overlap* to *independent* results in a drop of about 1% F1 absolute performance for the LB-MEM model with 5 and 10 memory cells, the other two models are almost unaffected. The reason why *overlap* is crucial to the LB-MEM model is because on average the tokens get more future context which helps the model in "anticipating" which entities are important and need to be kept in the memory.

A.3 Ground Truth Generation

In this section we explain how the ground truth action sequence is generated corresponding to the predicted mention sequence. The ground truth for U-MEM model is fairly straight forward. For the bounded memory models, we keep growing the number of entities till we hit the memory ceiling. For all the entities in memory, we maintain the number of mentions remaining in the ground truth cluster. For example, a cluster with a total of five mentions, two of which have already been processed by the model, has three remaining mentions.

Suppose now a mention corresponding to a currently untracked entity comes in and the memory is already at full capacity. Then for the LB-MEM model, we compare the number of mentions of this new entity (along with the current mention) against the number of mentions remaining for all the entities currently being tracked. If there are entities in memory with number of remaining mentions less than or equal to the number of mentions of this currently untracked entity, then the untracked entity replaces the entity with the least number of remaining mentions. Ties among the entities with least number of remaining mentions are broken by the least recently seen entity. If there's no such entity in the memory, then the mention is ignored. For the RB-MEM model, the comparison is done in a similar way but is limited to the LRU entity.

A.4 Miscellaneous

Computing Infrastructure & Runtime All the models for a single cross validation split of LitBank can be trained within 4 hours. The U-MEM models require 24GB memory GPUs and are trained on TitanRTX. The LB-MEM and RB-MEM models can be trained on 12GB memory GPUs.

As in LitBank, the U-MEM model for OntoNotes require 24GB memory GPUs. The LB-MEM and RB-MEM models can be trained on 12GB memory GPUs. Training on OntoNotes finishes within 12 hours.

Table 9: Number of model parameters (in millions).

	LitBank	OntoNotes
U-MEM	37.36	37.42
LB-MEM	46.83	46.95
RB-MEM	46.83	46.95

Table 10: Spearman correlation of F1 score with document length and # of entities in OntoNotes dev set.

Model	Document Length	# of Entities
U-MEM	-0.31	-0.27
LB-MEM		
5 cells	-0.38	-0.39
10 cells	-0.37	-0.35
20 cells	-0.30	-0.27
RB-MEM		
5 cells	-0.42	-0.47
10 cells	-0.36	-0.37
20 cells	-0.33	-0.30

Number of model parameters. Table 9 shows the number of trainable parameters for all the model and dataset combinations. LB-MEM and RB-MEM have additional parameters in comparison to U-MEM for predicting a score corresponding to the number of remaining mentions for an entity. Comparing across datasets, the OntoNotes models have a few additional parameters than their LitBank counterparts for modeling the document genre.

Evaluation Metric Code. We use the coreference scorer Perl script available at `https://github.com/conll/reference-coreference-scorers`. We also use the Python implementation by Kenton Lee available at `https://github.com/kentonl/e2e-coref/blob/master/metrics.py`. The two scripts can have some rounding differences.

Effect of Document Length and Number of Entities. Table 10 presents the Spearman correlation between document F1 score and both document length and number of entities in the document. The correlations are negative because the problem becomes more challenging with increase in document length and entities. The increase in memory for bounded models results in less negative correlation, suggesting improved performance for challenging documents. The slightly less negative correlation for LB-MEM models than RB-MEM models for 20 memory cells (when their dev performance is similar) implies that LB-MEM models perform better for longer OntoNotes documents.