Coreference Resolution without Span Representations

Yuval Kirstain* **Ori Ram*** **Omer Levy**

Blavatnik School of Computer Science, Tel Aviv University
{yuval.kirstain,ori.ram,levyomer}@cs.tau.ac.il

Abstract

The introduction of pretrained language models has reduced many complex task-specific NLP models to simple lightweight layers. An exception to this trend is coreference resolution, where a sophisticated task-specific model is appended to a pretrained transformer encoder. While highly effective, the model has a very large memory footprint – primarily due to dynamically-constructed span and span-pair representations – which hinders the processing of complete documents and the ability to train on multiple instances in a single batch. We introduce a lightweight end-to-end coreference model that removes the dependency on span representations, handcrafted features, and heuristics. Our model performs competitively with the current standard model, while being simpler and more efficient.

1 Introduction

Until recently, the standard methodology in NLP was to design task-specific models, such as BiDAF for question answering (Seo et al., 2017) and ESIM for natural language inference (Chen et al., 2017). With the introduction of pretraining, many of these models were replaced with simple output layers, effectively fine-tuning the transformer layers below to perform the traditional model's function (Radford et al., 2018). A notable exception to this trend is *coreference resolution*, where a multi-layer task-specific model (Lee et al., 2017, 2018) is appended to a pretrained model (Joshi et al., 2019, 2020). This model uses intricate span and span-pair representations, a representation refinement mechanism, handcrafted features, pruning heuristics, and more. While the model is highly effective, it comes at a great cost in memory consumption, limiting the amount of examples that can be loaded on a large GPU to a single document, which often needs to be truncated or processed in sliding windows. Can this coreference model be simplified?

We present *start-to-end* (s2e) coreference resolution: a simple coreference model that does *not* construct span representations. Instead, our model propagates information to the span boundaries (i.e., its start and end tokens) and computes mention and antecedent scores through a series of bilinear functions over their contextualized representations. Our model has a significantly lighter memory footprint, allowing us to process multiple documents in a single batch, with no truncation or sliding windows. We do not use any handcrafted features, priors, or pruning heuristics.

Experiments show that our minimalist approach performs on par with the standard model, despite removing a significant amount of complexity, parameters, and heuristics. Without any hyperparameter tuning, our model achieves 80.3 F1 on the English OntoNotes dataset (Pradhan et al., 2012), with the best comparable baseline reaching 80.2 F1 (Joshi et al., 2020), while consuming less than a third of the memory. These results suggest that transformers can learn even difficult structured prediction tasks such as coreference resolution without investing in complex task-specific architectures.[1]

2 Background: Coreference Resolution

Coreference resolution is the task of clustering multiple mentions of the same entity within a given text. It is typically modeled by identifying entity mentions (contiguous spans of text), and predicting an *antecedent* mention a for each span q (query) that refers to a previously-mentioned entity, or a null-span ε otherwise.

Lee et al. (2017, 2018) introduce *coarse-to-fine* (c2f), an end-to-end model for coreference resolu-

*Equal contribution.

[1]Our code and model are publicly available: https://github.com/yuvalkirstain/s2e-coref

tion that predicts, for each span q, an antecedent probability distribution over the candidate spans c:

$$P\left(a = c|q\right) = \frac{\exp(f(c,q))}{\sum_{c'} \exp(f(c',q))}$$

Here, $f(c,q)$ is a function that scores how likely c is to be an antecedent of q. This function is comprised of mention scores $f_m(c), f_m(q)$ (i.e. is the given span a mention?) and a separate antecedent score $f_a(c,q)$:

$$f(c,q) = \begin{cases} f_m(c) + f_m(q) + f_a(c,q) & c \neq \varepsilon \\ 0 & c = \varepsilon \end{cases}$$

Our model (Section 3) follows the scoring function above, but differs in how the different elements $f_m(\cdot)$ and $f_a(\cdot)$ are computed. We now describe how f_m and f_a are implemented in the c2f model.

Scoring Mentions In the c2f model, the mention score $f_m(q)$ is derived from a vector representation $\mathbf{v}_q$ of the span q (analogously, $f_m(c)$ is computed from $\mathbf{v}_c$). Let $\mathbf{x}_i$ be the contextualized representation of the i-th token produced by the underlying encoder. Every span representation is a concatenation of four elements: the representations of the span's start and end tokens $\mathbf{x}_{q_s}, \mathbf{x}_{q_e}$, a weighted average of the span's tokens $\hat{\mathbf{x}}_q$ computed via self-attentive pooling, and a feature vector $\phi(q)$ that represents the span's length:

$$\mathbf{v}_q = [\mathbf{x}_{q_s}; \mathbf{x}_{q_e}; \hat{\mathbf{x}}_q; \phi(q)]$$

The mention score $f_m(q)$ is then computed from the span representation $\mathbf{v}_q$:

$$f_m(q) = \mathbf{v}_m \cdot \text{ReLU}(\mathbf{W}_m \mathbf{v}_q)$$

where $\mathbf{W}_m$ and $\mathbf{v}_m$ are learned parameters. Then, span representations are enhanced with more global information through a refinement process that interpolates each span representation with a weighted average of its candidate antecedents. More recently, Xu and Choi (2020) demonstrated that this span refinement technique, as well as other modifications to it (e.g. entity equalization (Kantor and Globerson, 2019)) do not improve performance.

Scoring Antecedents The antecedent score $f_a(c,q)$ is derived from a vector representation of the span *pair* $\mathbf{v}_{(c,q)}$. This, in turn, is a function of the individual span representations $\mathbf{v}_c$ and $\mathbf{v}_q$, as well as a vector of handcrafted features $\phi(c,q)$ such as the distance between the spans c and q, the document's genre, and whether c and q were said/written by the same speaker:

$$\mathbf{v}_{(c,q)} = [\mathbf{v}_c; \mathbf{v}_q; \mathbf{v}_c \circ \mathbf{v}_q; \phi(c,q)]$$

The antecedent score $f_a(c,q)$ is parameterized with $\mathbf{W}_a$ and $\mathbf{v}_a$ as follows:

$$f_a(c,q) = \mathbf{v}_a \cdot \text{ReLU}(\mathbf{W}_a \mathbf{v}_{(c,q)})$$

Pruning Holding the vector representation of every possible span in memory has a space complexity of $O(n^2 d)$ (where n is the number of input tokens, and d is the model's hidden dimension). This problem becomes even more acute when considering the space of span *pairs* ($O(n^4 d)$). Since this is not feasible, candidate mentions and antecedents are pruned through a variety of model-based and heuristic methods.

Specifically, mention spans are limited to a certain maximum length ℓ. The remaining mentions are then ranked according to their scores $f_m(\cdot)$, and only the top λn are retained, while avoiding overlapping spans. Antecedents (span pairs) are further pruned using a lightweight antecedent scoring function (which is added to the overall antecedent score), retaining only a constant number of antecedent candidates c for each target mention q.

Training For each remaining span q, the training objective optimizes the marginal log-likelihood of all of its unpruned gold antecedents c, as there may be multiple mentions referring to the same entity:

$$\log \sum_{c} P(a = c|q)$$

Processing Long Documents Due to the c2f model's high memory consumption and the limited sequence length of most pretrained transformers, documents are often split into segments of a few hundred tokens each (Joshi et al., 2019). Recent work on efficient transformers (Beltagy et al., 2020) has been able to shift towards processing complete documents, albeit with a smaller model (base) and only one training example per batch.

3 Model

We present *start-to-end* (s2e) coreference resolution, a simpler and more efficient model with respect to c2f (Section 2). Our model utilizes the endpoints of a span (rather than all span tokens) to compute the mention and antecedent scores $f_m(\cdot)$

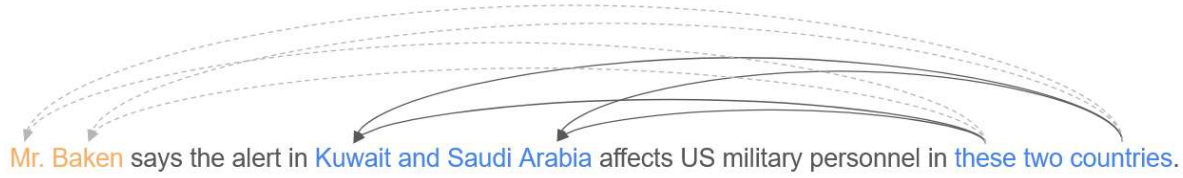

Figure 1: The antecedent score $f_a(c, q)$ of a query mention $q = (q_s, q_e)$ and a candidate antecedent $c = (c_s, c_e)$ is defined via bilinear functions over the representations of their endpoints c_s, c_e, q_s, q_e. Solid lines reflect factors participating in positive examples (coreferring mentions), and dashed lines correspond to negative examples.

and $f_a(\cdot, \cdot)$ *without* constructing span or span-pair representations; instead, we rely on a combination of lightweight bilinear functions between pairs of endpoint token representations. Furthermore, our model does not use any handcrafted features, does not prune antecedents, and prunes mention candidates solely based on their mention score $f_m(q)$.

Our computation begins by extracting a *start* and *end* token representation from the contextualized representation $\mathbf{x}$ of each token in the sequence:

$$\mathbf{m}^s = \text{GeLU}(\mathbf{W}_m^s \mathbf{x}) \qquad \mathbf{m}^e = \text{GeLU}(\mathbf{W}_m^e \mathbf{x})$$

We then compute each mention score as a biaffine product over the start and end tokens' representations, similar to Dozat and Manning (2017):

$$f_m(q) = \mathbf{v}_s \cdot \mathbf{m}_{q_s}^s + \mathbf{v}_e \cdot \mathbf{m}_{q_e}^e + \mathbf{m}_{q_s}^s \cdot \mathbf{B}_m \cdot \mathbf{m}_{q_e}^e$$

The first two factors measure how likely the span's start/end token q_s/q_e is a beginning/ending of an entity mention. The third measures whether those tokens are the boundary points of the *same* entity mention. The vectors $\mathbf{v}_s, \mathbf{v}_e$ and the matrix $\mathbf{B}_m$ are the trainable parameters of our mention scoring function f_m. We efficiently compute mention scores for all possible spans while masking spans that exceed a certain length ℓ.[2] We then retain only the top-scoring λn mention candidates to avoid $O(n^4)$ complexity when computing antecedents.

Similarly, we extract *start* and *end* token representations for the antecedent scoring function f_a:

$$\mathbf{a}^s = \text{GeLU}(\mathbf{W}_a^s \mathbf{x}) \qquad \mathbf{a}^e = \text{GeLU}(\mathbf{W}_a^e \mathbf{x})$$

Then, we sum over four bilinear functions:

$$\begin{aligned} f_a(c, q) = \mathbf{a}_{c_s}^s \cdot \mathbf{B}_a^{ss} \cdot \mathbf{a}_{q_s}^s + \mathbf{a}_{c_s}^s \cdot \mathbf{B}_a^{se} \cdot \mathbf{a}_{q_e}^e \\ + \mathbf{a}_{c_e}^e \cdot \mathbf{B}_a^{es} \cdot \mathbf{a}_{q_s}^s + \mathbf{a}_{c_e}^e \cdot \mathbf{B}_a^{ee} \cdot \mathbf{a}_{q_e}^e \end{aligned}$$

Each component measures the compatibility of the spans c and q by an interaction between different boundary tokens of each span. The first component compares the *start* representations of c and q, while the fourth component compares the *end* representations. The second and third facilitate a cross-comparison of the *start* token of span c with the *end* token of span q, and vice versa. Figure 1 (bottom) illustrates these interactions.

This calculation is equivalent to computing a bilinear transformation between the concatenation of each span's boundary tokens' representations:

$$f_a(c, q) = [\mathbf{a}_{c_s}^s; \mathbf{a}_{c_e}^e] \cdot \mathbf{B}_a \cdot [\mathbf{a}_{q_s}^s; \mathbf{a}_{q_e}^e]$$

However, computing the factors *directly* bypasses the need to create n^2 explicit span representations. Thus, we avoid a theoretical space complexity of $O(n^2 d)$, while keeping it equivalent to that of a transformer layer, namely $O(n^2 + nd)$.

4 Experiments

Dataset We train and evaluate on two datasets: the document-level English OntoNotes 5.0 dataset (Pradhan et al., 2012), and the GAP coreference dataset (Webster et al., 2018). The OntoNotes dataset contains speaker metadata, which the baselines use through a hand-crafted feature that indicates whether two spans were uttered by the same speaker. Instead, we insert the speaker's name to the text every time the speaker changes, making the metadata available to any model.

Pretrained Model We use Longformer-Large (Beltagy et al., 2020) as our underlying pretrained model, since it is able to process long documents without resorting to sliding windows or truncation.

Baseline We consider Joshi et al.'s (2019) expansion to the c2f model as our baseline. Specifically, we use the implementation of Xu and Choi (2020) with minor adaptations for supporting Longformer. We do not use higher-order inference, as Xu and Choi (2020) demonstrate that it does not result in significant improvements. We train the baseline

[2] While pruning by length is not necessary for efficiency, we found it to be a good inductive bias.

Model	MUC			B^3			$CEAF_{\phi_4}$			
	P	R	F1	P	R	F1	P	R	F1	Avg. F1
c2f + SpanBERT-Large	85.7	**85.3**	85.5	79.5	**78.7**	**79.1**	**76.8**	75.0	75.9	80.2
c2f + Longformer-Base	85.0	85.0	85.0	77.8	77.8	77.8	75.6	74.2	74.9	79.2
c2f + Longformer-Large	86.0	83.2	84.6	78.9	75.5	77.2	76.7	68.7	72.5	78.1
s2e + Longformer-Large	**86.5**	85.1	**85.8**	**80.3**	77.9	**79.1**	**76.8**	**75.4**	**76.1**	**80.3**

Table 1: Performance on the test set of the English OntoNotes 5.0 dataset. *c2f* refers to the course-to-fine approach of Lee et al. (2017, 2018), as ported to pretrained transformers by Joshi et al. (2019).

	Masc	**Fem**	**Bias**	**Overall**
c2f + SpanBERT-Large	90.5	**86.3**	**0.95**	**88.4**
c2f + Longformer-Base	87.6	82.3	0.94	84.9
c2f + Longformer-Large	90.1	85.4	**0.95**	87.8
s2e + Longformer-Large	**90.6**	85.8	**0.95**	88.3

Table 2: Performance on the test set of the GAP coreference dataset. The reported metrics are F1 scores.

Model	Memory (GB)
c2f + SpanBERT-Large	16.2
c2f + Longformer-Base	12.0
c2f + Longformer-Large	15.7
s2e + Longformer-Large	**4.3**

Table 3: Peak GPU memory usage during inference on OntoNotes, when processing one document at a time.

model over three pretrained models: Longformer-Base, Longformer-Large, and SpanBERT-Large (Beltagy et al., 2020; Joshi et al., 2020).

Hyperparameters All models use the same hyperparameters as the baseline. The only hyperparameters we change are the maximum sequence length and batch size, which we enlarge to fit as many tokens as possible into a 32GB GPU.[3] For our model, we use dynamic batching with 5,000 max tokens, which allows us to fit an average of 5-6 documents in every training batch. The baseline, however, has a much higher memory footprint, and is barely able to fit a single example with Longformer-Base (max 4,096 tokens). When combining the baseline with SpanBERT-Large or Longformer-Large, the baseline must resort to sliding windows to process the full document (512 and 2,048 tokens, respectively).

Performance Table 1 and Table 2 show that, despite our model's simplicity, it performs as well as the best performing baseline. Our model with Longformer-Large achieves 80.3 F1 on OntoNotes, while the best performing baseline achieves 80.2 F1. When the baseline model is combined with either version of Longformer, it is not able to reach the same performance level as our model. We see similar trends for GAP. Our findings indicate that there is little to lose from simplifying the coreference resolution architecture, while there are potential gains to be had from optimizing with larger batches.

Efficiency We also compare our model's memory usage using the OntoNotes development set. Table 3 shows that our implementation is at least three times more memory efficient than the baseline. This improvement results from a combination of three factors: (1) the fact that our model is lighter on memory and does not need to construct span or span-pair representations, (2) our simplified framework, which does not use sliding windows, and (3) our implementation, which was written "from scratch", and might thus be more (or less) efficient than the original.

5 Related Work

Recent work on memory-efficient coreference resolution sacrifices speed and parallelism for guarantees on memory consumption. Xia et al. (2020) and Toshniwal et al. (2020) present variants of the c2f model (Lee et al., 2017, 2018) that use an iterative process to maintain a fixed number of span representations at all times. Specifically, spans are processed sequentially, either joining existing clusters or forming new ones, and an eviction mechanism ensures the use of a constant number of clusters. While these approach constrains the space complexity, their sequential nature slows down the computation, and slightly deteriorates the performance. Our approach is able to alleviate the large

[3] We made one exception, and tried to tune the Longformer-Large baseline's hyperparameters. Despite our efforts, it still performs worse than Longformer-Base.

memory footprint of c2f while maintaining fast parallel processing and high performance.

CorefQA (Wu et al., 2020) propose an alternative solution by casting the task of coreference resolution as one of extractive question answering. It first detects potential mentions, and then creates dedicated queries for each one, creating a pseudo-question-answering instance for each candidate mention. This method significantly improves performance, but at the cost of processing hundreds of individual context-question-answer instances for a single document, substantially increasing execution time. Our work provides a simple alternative, which can scale well in terms of both speed and memory.

6 Conclusion

We introduce a new model for coreference resolution, suggesting a lightweight alternative to the sophisticated model that has dominated the task over the past few years. Our model is competitive with the baseline, while being simpler and more efficient. This finding once again demonstrates the spectacular ability of deep pretrained transformers to model complex natural language phenomena.

Acknowledgements

This research was funded by the Blavatnik Fund, the Alon Scholarship, Intel Corporation, and the Yandex Initiative in Machine Learning.

References

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

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668, Vancouver, Canada. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *ICLR 2017*.

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.

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

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

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

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.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. *ArXiv*, abs/1611.01603.

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

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.

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.

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.