Separately Parameterizing Singleton Detection Improves End-to-end Neural Coreference Resolution

Xiyuan Zou [1 2], **Yiran Li** [1], **Ian Porada** [1 2], **Jackie Chi Kit Cheung** [1 2]
[1] McGill University, [2] Mila Quebec AI Institute
{xiyuan.zou, yiran.li3, ian.porada}@mail.mcgill.ca, jackie.cheung@mcgill.ca

Abstract

Current end-to-end coreference resolution models combine detection of singleton mentions and antecedent linking into a single step. In contrast, singleton detection was often treated as a separate step in the pre-neural era. In this work, we show that separately parameterizing these two sub-tasks also benefits end-to-end neural coreference systems. Specifically, we add a singleton detector to the coarse-to-fine (C2F) coreference model, and design an anaphoricity-aware span embedding and singleton detection loss. Our method significantly improves model performance on OntoNotes and four additional datasets.[1]

1 Introduction

Coreference resolution (CR) is the task of identifying and clustering linguistic expressions that refer to the same real-world entity. Recent progress in CR has been led by various end-to-end (e2e) neural models (Lee et al., 2017, 2018; Joshi et al., 2019; Kirstain et al., 2021; Otmazgin et al., 2023) which significantly outperform older pipelined systems. Many of these e2e models follow the design of Lee et al. (2017), jointly training both a mention detector that extracts candidate mentions from all text spans and a mention linker that assigns the antecedent to each candidate mention. Despite their impressive performance, these e2e CR models are far from perfect: replacing either the mention detector or linker with an oracle results in a substantial improvement of the entire model (Wu and Gardner, 2021). This indicates room for improving the mention detector and linker in the current joint systems.

Indeed, modeling CR as mention detection followed by mention linking is not a clear decomposition because mention linking itself is composed of two sub-tasks: singleton detection and antecedent linking. Singletons are mentions that refer to entities which only appear once in the discourse and are often removed from model predictions because they are not coreferring. It is important to correctly distinguish anaphoric mentions from singletons since singletons account for the majority of mentions: over 80% of mentions in the development set of OntoNotes are singletons (De Marneffe et al., 2015). Nevertheless, prior work shows that current mention detectors lack the ability to make such anaphoricity decisions (Wu and Gardner, 2021). Thus, the mention linker in current joint systems performs two tasks: it not only links anaphora with antecedents, but also identifies singletons by linking them to the empty antecedent. Singleton detection and antecedent linking, however, are two disparate tasks that may require different representations and relying on a single module hurts their performance. Wu and Gardner (2021) further note that the mention linker increases its confidence in assigning coreference scores when not tasked with singleton detection.

Incorporating an extra singleton detector is a straightforward solution and has been extensively investigated in the pre-neural era for pipelined systems (Recasens et al., 2013; De Marneffe et al., 2015; Moosavi and Strube, 2016). In this work, we show that it is also effective for neural end-to-end CR models. We extend the coarse-to-fine (C2F) model (Lee et al., 2018) by adding a separately parameterized singleton detector between the mention detector and linker. The singleton detector takes in the top-scoring candidate mentions extracted by the mention detector and predicts a singleton score for each candidate mention. Candidate mentions with the highest singleton scores are pruned out before being fed into the mention linker.

It is notable that the anaphoricity decision is more challenging than the mention decision because the former requires not only the information from the mention itself but also contextual clues

[1] Our code is available at https://github.com/XiyuanZou/C2F-SD

Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers), pages 212–219
June 16-21, 2024 ©2024 Association for Computational Linguistics

from its potential antecedents. We concatenate the mention embedding with an anaphoricity-aware span embedding produced from a span-level attention that explicitly attends to itself and all of its preceding candidate mentions. In addition, we add a singleton detection loss to explicitly supervise the singleton detector during the joint training. We show in Sec. 4 that these components are necessary, and there is little performance gain without them.

Zhong and Chen (2021) show that using separate encoders for each sub-task greatly improves the overall task of entity and relation extraction. Inspired by their work, we set up an expert representation learner for each of the mention detection, singleton detection, and antecedent linking tasks. In addition, we add a shared representation learner between these three sub-tasks as these sub-tasks are related and may benefit from certain shared features during training.

The overall architecture of our model is shown in Figure 1. Although C2F is used as the base, our method is general and can be applied to any CR model that follows the mention detector-linker architecture. We show that our model gains significant improvements on OntoNotes and four additional datasets compared to the base model and achieves a new SoTA among all detector-linker models. We also scale up our model to 2B parameters, outperforming the 11B ASP model (Liu et al., 2022) and approaching the current SoTA seq2seq model (Bohnet et al., 2023) while being considerably smaller in size and faster at inference.

2 Background: The C2F Model

In this section, we introduce the C2F model (Lee et al., 2018), which is one of the first e2e neural CR models. It was later outperformed by the LingMess model (Otmazgin et al., 2023) which is the current best model that follows the mention detector-linker architecture. Recently proposed seq2seq approaches (Liu et al., 2022; Bohnet et al., 2023; Zhang et al., 2023) also achieve substantially higher accuracy, but they require significantly more resources and have slower inference speed. We will show that with our method of separately parameterizing a singleton detector, the old C2F model can significantly outperform the LingMess model and narrow the gap with the SoTA seq2seq model.

The C2F model computes a span embedding v_q for each text span q. Let x_i be the contextual representation of the ith token produced by the

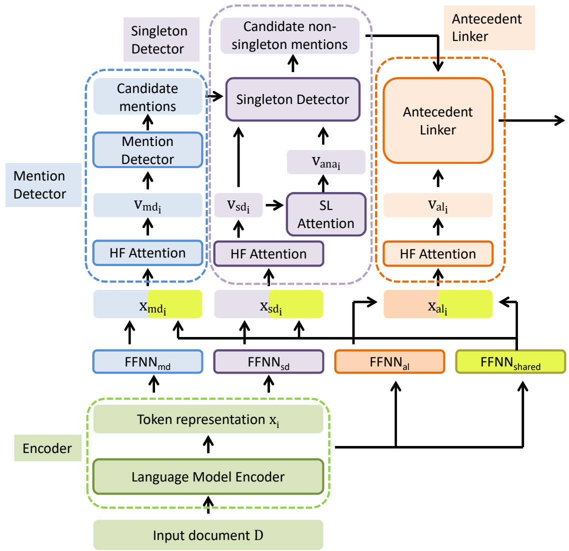

Figure 1: A high-level overview of our model. HF attention refers to the head-finding attention proposed by Lee et al. (2017) and SL attention is the span-level attention to make anaphoricity-aware embeddings.

LLM encoder. Each span embedding consists of the representation of the start token x_{q_s}, the end token x_{q_e}, and the weighted average of all tokens within the span $\hat{x}_q$ computed via a head-finding attention, and a feature vector ϕ_q encoding the span width:

$$v_q = [x_{q_s}; x_{q_e}; \hat{x}_q; \phi_q]$$

C2F first consists of a mention detector, which is essentially a feed forward network that computes a mention score s_m for each span based on the span embedding:

$$s_m(q) = \text{FFNN}_m(v_q)$$

Those spans with the highest mention scores are retained as candidate mentions. For each candidate mention i, C2F roughly scores the possible antecedents of i by a lightweight bilinear function and keeps a constant number of top-scoring spans as its candidate antecedents $\mathcal{Y}(i)$. The antecedent linker, which is another feed forward network, then computes an antecedent score s_a between i and each of its candidate antecedents j:

$$s_a(i, j) = \text{FFNN}_a([v_i; v_j; v_i \odot v_j; \phi_{ij}])$$

where ϕ_{ij} is a vector of pairwise features such as the distance between spans, whether two spans are from the same speaker, etc. The final pairwise coreference score $s(i, j)$ is the sum of the mention scores and the antecedent score:

$$s(i,j) = \begin{cases} s_m(i) + s_m(j) + s_a(i,j), & j \neq \epsilon \\ 0, & j = \epsilon \end{cases}$$

where ϵ is the empty antecedent. Finally, C2F predicts an antecedent distribution for each candidate mention i:

$$P(a = j \mid i) = \frac{\exp\left(s(i,j)\right)}{\Sigma_{j' \in \mathcal{Y}(i)} \exp\left(s(i,j')\right)}$$

During training, C2F optimizes the marginal log-likelihood of each candidate mention i being assigned all of its unpruned gold antecedents $j \in \mathcal{Y}(i) \cap \text{Gold}(i)$:

$$L_{\text{Coref}} = -\log \prod_i \sum_{j \in \mathcal{Y}(i) \cap \text{Gold}(i)} P(a = j \mid i)$$

3 Methodology

Our core contribution is to add a singleton detector to the C2F architecture. To exploit the similarities and differences between distinct sub-tasks of CR, we build an expert representation learner for each of the mention detector (md), singleton detector (sd), and antecedent linker (al) and also a general representation learner shared between them. The representation of each token x_i for each sub-task t is the concatenation of the expert and the shared representation:

$$x_{\text{share}_i} = \text{FFNN}_{\text{share}}(x_i)$$

$$x_{t_i} = [\text{FFNN}_t(x_i); x_{\text{share}_i}], \quad t \in \{\text{md, sd, al}\}$$

We follow the same approach as C2F to create a span embedding v_{t_q} for each sub-task t. Additionally, we make an anaphoricity-aware embedding to improve the ability of the singleton detector to make anaphoricity decisions. For this, we use additive attention (Bahdanau et al., 2015), but applied on the span-level where each candidate mention i attends to itself and all of its preceding unpruned candidate mentions:

$$f_{\text{att}}(i,j) = w_v{}^{\intercal} \tanh\left(W_q v_{\text{sd}_i} + W_k v_{\text{sd}_j}\right)$$

$$\alpha_{ij} = \frac{\exp\left(f_{\text{att}}(i,j)\right)}{\sum_{j' \in i \cup \text{Preceding}(i)} \exp\left(f_{\text{att}}(i,j')\right)}$$

$$v_{\text{ana}_i} = \sum_{j \in i \cup \text{Preceding}(i)} \alpha_{ij} \cdot v_{\text{sd}_j}$$

The singleton detector computes a singleton score s_s for each candidate mention using both the anaphoricity-aware embedding and the original span embedding:

$$s_s(i) = \text{FFNN}_s([v_{\text{ana}_i}; v_{\text{sd}_i}])$$

The top K percentile of spans with highest singleton scores are identified as singletons and pruned out. We keep the antecedent linker unchanged and the final pairwise coreference score $s(i,j)$ now becomes the sum of the mention scores and the antecedent score minus the singleton scores.

$$s(i,j) = \begin{cases} s_m(i) + s_m(j) + s_a(i,j) - s_s(i) - s_s(j), & j \neq \epsilon \\ 0, & j = \epsilon \end{cases}$$

We further introduce a singleton detection loss to explicitly supervise the singleton detector:

$$L_{\text{Singleton}} = -\sum_i \mathbb{1}(i) \log(1 - S_s(i)) + (1 - \mathbb{1}(i)) \log(S_s(i))$$

where $\mathbb{1}(i)$ is an indicator function that equals to 1 if the span i is a gold non-singleton mention and 0 otherwise. This is essentially a binary cross entropy loss that pushes down the singleton scores of those coreferent mentions and pushes up the scores of all singletons. Our final objective is a weighted sum of the coreference loss and the singleton detection loss:

$$L = \lambda_1 L_{\text{Coref}} + \lambda_2 L_{\text{Singleton}}$$

4 Experiments

Dataset We train and evaluate on the OntoNotes 5.0 English dataset (Weischedel et al., 2013) and four additional datasets: WiKiCoref (Ghaddar and Langlais, 2016), OntoGUM (Zhu et al., 2021b), GAP (Webster et al., 2018) and WinoBias (Zhao et al., 2018). These datasets do not annotate singletons and thus require models to filter out any potential singletons.

Baseline We re-implement and re-train the C2F model (Lee et al., 2018) as a baseline and build our model upon it. The original C2F model comes with a higher-order inference step which we do not include as it marginally affects performance (Xu and Choi, 2020). We also re-implement the recently developed LingMess model (Otmazgin et al., 2023) as a stronger baseline. In addition, we compare our model to the ASP (Liu et al., 2022) at 11B and the Link-Append model at 13B parameters (Bohnet et al., 2023). Unfortunately, we do not have enough resources to train these large seq2seq

	MUC			B^3			CEAFϕ_4			
	R	P	F1	R	P	F1	R	P	F1	Avg F1
LingMess	84.6	**88.2**	86.3	78.3	83.1	80.7	76.3	78.1	77.2	81.4
C2F	85.2	86.5	85.9	**79.0**	80.2	79.6	76.4	76.6	76.5	80.7
C2F + singleton detector	**85.4**	88.0	**86.7**	78.8	**83.5**	**81.1**	**76.9**	**79.2**	**78.1**	**81.9**

Table 1: Model performance on the test set of the OntoNotes 5.0 English dataset measured by the CoNLL F1 score averaged from MUC, B^3, CEAFϕ_4. Our approach of separately parameterizing a singleton detector achieves an increase that is statistically significant according to a non-parametric permutation test ($p < 0.05$).

	C2F	C2F + singleton detector
WiKiCoref	61.2	**63.0**
OntoGUM	67.7	**68.6**
GAP	88.9	**89.8**
WinoBias	84.5	**85.3**

Table 2: Model performance on the test set of 4 additional CR datasets. WiKiCoref and OntoGUM are evaluated by CoNLL F1 score, GAP by F1 score and WinoBias by accuracy.

	LM	Avg F1	Size	Time
C2F + SD	DeBERTa-xxl	82.6	**2.0B**	**637.4**
Link-Append	mT5-xxl	**83.3**	13B	6.0e5
ASP	FlanT5-xxl	82.5	11B	N/A

Table 3: Comparison between our model and the SOTA seq2seq models after scaling up. Inference is done on OntoNotes test set using a single 80 GB A100 GPU. Model performance is measured by CoNLL F1 score and time is inference speed (ms/doc) at max batch size.

models. For Link-Append, we load the publicly released weights. For the ASP model, we compare against the reported results as finetuned weights are not available.

Pretrained Encoder We use DeBERTa-large (He et al., 2020) as the pretrained encoder for the C2F baseline and our model since DeBERTa outperforms other pretrained encoder models for CR (Porada et al., 2024). To compete with seq2seq models that are considerably larger, we scale up our model by using DeBERTa-v2-xxl.

Main Results Table 1 and 2 show that our method improves the C2F base model by 1.2 absolute points on OntoNotes, 1.8 on WikiCoref, 1.1 on OntoGUM, 0.9 on GAP and 0.8 on WinoBias. All of these performance increases are statistically significant, showing the effectiveness of separately parameterizing a singleton detector in CR systems. Our model also outperforms the LingMess model by 0.5 on OntoNotes and achieves a new SoTA among all detector-linker CR models. Table 3 further shows that our model at 2B parameter size outperforms the 11B ASP. Although there is still a gap of 0.7 to the 13B Link-Append model, our model is about 6.5 times smaller and 95 times faster in inference speed, thus more practical to use.

Importance of Singleton Detector To assess that the improvement of our model is due to the independent parameterization of singleton detection rather than the added parameters, we increase the parameter count of the original C2F model by adding extra layers to its mention linker to match the number of parameters of our model. We observe that simply adding more parameters to the mention linker without separately parameterizing singleton detection surprisingly results in a 0.2 absolute drop of CoNLL F1 score on OntoNotes.

Importance of Anaphoricity-aware Span Embedding and Singleton Detection Loss
We find that the anaphoricity-aware span embedding together with the singleton detection loss is important to the success of the singleton detector. To show this, we perform a series of ablation studies on OntoNotes (table 4). Firstly, we concatenate the mention embedding with a copy of itself rather than the anaphoricity-aware embedding, leading to a 1.2 decrease in model performance, reducing it to the same accuracy as the original C2F model. Secondly, we train a model without $L_{\text{singleton}}$ in which case we observe a 0.9 absolute drop of CoNLL F1 score. In addition, we independently ablate the shared and the expert representation learners. In both cases, the performance witnesses a statistically significant drop, but not as much as when ablating the anaphoricity-aware span embedding and the singleton detection loss.

Singleton Detector Imposes Heavier Penalties on Singletons than on Non-entity Spans
To better understand the model behavior, we count the average number of non-entity spans, coreferent

	Avg F1	Δ
C2F + SD	81.9	–
w/o anaphoricity-aware embedding	80.7	-1.2
w/o singleton detection loss	81.0	-0.9
w/o shared representation learner	81.5	-0.4
w/o expert representation learners	81.4	-0.5

Table 4: Ablation studies for each proposed module of the C2F+SD model on the test set of the OntoNotes 5.0 English dataset measured by the CoNLL F1 score.

	Before MD		
	Non-entity	Coreferent	Singletons
C2F Base	5036.99	51.92	51.91
C2F + SD	5036.99	51.92	51.91
	After MD (Before SD)		
	Non-entity	Coreferent	Singletons
C2F Base	128.38	45.52	41.62
C2F + SD	135.44	47.13	44.69
	After SD		
	Non-entity	Coreferent	Singletons
C2F Base	–	–	–
C2F + SD	121.25	45.90	15.66

Table 5: The average number of non-entity spans, coreferent spans and singletons per document at each processing stage of the original C2F model and the C2F+SD model. MD and SD stand for mention detector and singleton detector respectively. Models are trained on the OntoNotes 5.0 English dataset and tested on the test set of PreCo (Chen et al., 2018).

spans and singletons per document at each processing stage of the original C2F model and our C2F+SD model. Counting singletons requires gold annotation of singletons. Thus we test the models on PreCo (Chen et al., 2018), where singletons are annotated. As shown in table 5, we find that 99.4% spans filtered by the mention detector of the original C2F model are non-entity spans. There are still over 80% singletons left and the mention detector does not have the ability to filter out these singletons. In our C2F+SD model, 65.3% spans filtered out by the singleton detector are the singletons, and only 30.2% singletons remain after singleton detection compared to 86.1% before it. In addition, we observe that among the remaining spans, on average, the singleton score for singletons is 279% higher than that for non-entity spans and 46% higher than for coreferent spans. These results indicate that our design of the singleton detector imposes significant penalties on singletons, something that is absent in the original C2F model.

5 Related Work

Singleton detection has been extensively explored in the pre-neural era for the pipelined CR systems. Recasens et al. (2013) builds a logistic regression model with both surface (i.e. part-of-speech and n-gram based) features and carefully designed linguistic features for predicting the distinction between singletons and coreferent spans. They incorporate it into a SoTA CR pipeline and yield a significant performance improvement. Moosavi and Strube (2016) models singleton detection by an anchored SVM and use only a small set of shallow features to achieve similarly significant improvements across various CR models.

However, singleton detection still remains under-explored for end-to-end neural CR models. Zhu et al. (2023) design a multi-task learning based neural coreference model which learns singletons jointly with other tasks such as entity type recognition. Their model achieves SoTA results on OntoGUM (Zhu et al., 2021b) and generalizes robustly to two other datasets. It is notable that their approach assumes the gold annotations of entity types and information status which are not commonly annotated in many coreference datasets. As a comparison, our model does not require additional information beyond what the original C2F model requires.

6 Conclusion

We decouple the singleton detection and the antecedent linking in the current detector-linker CR models by separately parameterizing a singleton detector. The effectiveness of our method shows that a separate singleton detection step benefits neural end-to-end CR systems. This also points out a future research direction: how to build a stronger singleton detector in end-to-end systems.

7 Limitations

Separately parameterizing a singleton detector introduces extra parameters and increases the inference time and the memory usage. Moreover, we build our model around OntoNotes and other datasets where singletons are not annotated. On datasets where singletons are explicitly annotated, it is not clear if our proposed method will result in similar improvements as those observed in our experiments.

Acknowledgements

The authors acknowledge the material support of NVIDIA in the form of computational resources. Xiyuan Zou is supported by a McGill Science Undergraduate Research Award (SURA). Ian Porada is supported by a fellowship from the Fonds de recherche du Québec (FRQ). Jackie Chi Kit Cheung is supported by the Canada CIFAR AI Chair program.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Bernd Bohnet, Chris Alberti, and Michael Collins. 2023. Coreference resolution through a seq2seq transition-based system. *Transactions of the Association for Computational Linguistics*, 11:212–226.

Hong Chen, Zhenhua Fan, Hao Lu, Alan Yuille, and Shu Rong. 2018. PreCo: A large-scale dataset in preschool vocabulary for coreference resolution. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 172–181, Brussels, Belgium. Association for Computational Linguistics.

Marie-Catherine De Marneffe, Marta Recasens, and Christopher Potts. 2015. Modeling the lifespan of discourse entities with application to coreference resolution. *Journal of Artificial Intelligence Research*, 52:445–475.

William Falcon and The PyTorch Lightning team. 2019. PyTorch Lightning.

Abbas Ghaddar and Phillippe Langlais. 2016. WikiCoref: An English coreference-annotated corpus of Wikipedia articles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 136–142, Portorož, Slovenia. European Language Resources Association (ELRA).

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention.

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.

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.

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.

Tianyu Liu, Yuchen Eleanor Jiang, Nicholas Monath, Ryan Cotterell, and Mrinmaya Sachan. 2022. Autoregressive structured prediction with language models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 993–1005, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Nafise Sadat Moosavi and Michael Strube. 2016. Search space pruning: A simple solution for better coreference resolvers. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1005–1011, San Diego, California. Association for Computational Linguistics.

Shon Otmazgin, Arie Cattan, and Yoav Goldberg. 2023. LingMess: Linguistically informed multi expert scorers for coreference resolution. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2752–2760, Dubrovnik, Croatia. Association for Computational Linguistics.

Ian Porada, Xiyuan Zou, and Jackie Chi Kit Cheung. 2024. A controlled reevaluation of coreference resolution models.

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.

Marta Recasens, Marie-Catherine de Marneffe, and Christopher Potts. 2013. The life and death of discourse entities: Identifying singleton mentions. In *Proceedings of the 2013 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 627–633, Atlanta, Georgia. Association for Computational Linguistics.

Kellie Webster, Marta Recasens, Vera Axelrod, and Jason Baldridge. 2018. Mind the gap: A balanced corpus of gendered ambiguou. In *Transactions of the ACL*, page to appear.

Ralph Weischedel et al. 2013. Ontonotes release 5.0.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zhaofeng Wu and Matt Gardner. 2021. Understanding mention detector-linker interaction in neural coreference resolution. In *Proceedings of the Fourth Workshop on Computational Models of Reference, Anaphora and Coreference*, pages 150–157, 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.

Amir Zeldes. 2017. The GUM corpus: Creating multilayer resources in the classroom. *Language Resources and Evaluation*, 51(3):581–612.

Wenzheng Zhang, Sam Wiseman, and Karl Stratos. 2023. Seq2seq is all you need for coreference resolution. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 11493–11504, Singapore. 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.

Zexuan Zhong and Danqi Chen. 2021. A frustratingly easy approach for entity and relation extraction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 50–61, Online. Association for Computational Linguistics.

Yilun Zhu, Siyao Peng, Sameer Pradhan, and Amir Zeldes. 2023. Incorporating singletons and mention-based features in coreference resolution via multi-task learning for better generalization. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 121–130, Nusa Dua, Bali. Association for Computational Linguistics.

Yilun Zhu, Sameer Pradhan, and Amir Zeldes. 2021a. Anatomy of OntoGUM—Adapting GUM to the OntoNotes scheme to evaluate robustness of SOTA coreference algorithms. In *Proceedings of the Fourth Workshop on Computational Models of Reference, Anaphora and Coreference*, pages 141–149, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yilun Zhu, Sameer Pradhan, and Amir Zeldes. 2021b. OntoGUM: Evaluating contextualized SOTA coreference resolution on 12 more genres. 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 461–467, Online. Association for Computational Linguistics.

Appendix A Implementation Details

A.1 Hyperparameters

We use Pytorch Lightning (Falcon and The PyTorch Lightning team, 2019) and HuggingFace Transformers (Wolf et al., 2020) to implement our model. We generally use the same hyperparameters as the original C2F model with a few exceptions. We report these changes here. As our model is memory intensive, we randomly truncate documents to 6 segments for DeBERTa-large and 3 segments for DeBERTa-v2-xxl. We set the maximum segment length to 512 for each segment. We use the hidden size of 3072 for the extra singleton detector introduced. We filter out top 40% candidate mentions with highest singleton scores. We use 1.0 for $\lambda 1$ and 0.6 for $\lambda 2$ to prioritize the coreference loss over the singleton detection loss. The DeBERTa-large model is trained for 50 epochs on a single 80GB A100, and the training takes about 18 hours. The DeBERTa-v2-xxl model is trained for 75 epochs on 4 80GB A100 GPUs, and the training takes about 1 and half a day.

A.2 Evaluation

We use the official CoNLL coreference scorer[2] for evaluating on OntoNotes, OntoGUM and WiKiCoref. We use the official GAP scorer[3] for evaluating on GAP.

Appendix B Dataset Details

Ontonotes 5.0 (Weischedel et al., 2013) is the most common dataset for training and evaluating CR models. We specifically use the CoNLL-2012 Shared Task v4 dataset split (Pradhan et al., 2012). The train/validation/test splits are 1940/343/348 document parts, respectively. This dataset covers 7 genres of text including telephone conversations, broadcast conversations, broadcast news, magazine, newswire, pivot text and web blogs. Genre and speaker information is annotated in OntoNotes, so we use them when training and evaluating our model.

OntoGUM (Zhu et al., 2021b) is composed of the coreference annotations in the English language GUM corpus (Zeldes, 2017) transformed heurstically to follow OntoNotes annotation guidelines (Zhu et al., 2021a). This dataset covers 12 different text genres. We use both genre and speaker information to help our model. There are totally 168 documents in OntoGUM. We randomly split it into 148/10/10 as the train/validation/test splits.

GAP (Webster et al., 2018) consists of pronouns in English Wikipedia annotated for coreference with respect to two preceding noun phrase. We do not use genre and speaker information for this dataset as they are not available. The train/validation/test splits are 4000/908/4000 coreference-labeled pairs, respectively.

WinoBias (Zhao et al., 2018) contains Winograd-schema style sentences with entities corresponding to people referred by their occupation. There are 1580 sentences in the training set and another 1580 sentences in the test set. We randomly take half the sentences from the test set as our validation set. We do not consider genre and speaker when evaluating our model.

WiKiCoref (Ghaddar and Langlais, 2016) is a CR dataset where all documents are sourced from English Wikipedia. It is a relatively small dataset with 30 documents. We do not consider genre and speaker for this dataset.

[2] https://github.com/conll/reference-coreference-scorers

[3] https://github.com/google-research-datasets/gap-coreference