# F-COREF: Fast, Accurate and Easy to Use Coreference Resolution

**Shon Otmazgin**[1] **Arie Cattan**[1] **Yoav Goldberg**[1,2]
[1]Computer Science Department, Bar Ilan University
[2]Allen Institute for Artificial Intelligence
{shon711,arie.cattan,yoav.goldberg}@gmail.com

## Abstract

We introduce *fastcoref*, a python package for fast, accurate, and easy-to-use English coreference resolution. The package is pip-installable, and allows two modes: an accurate mode based on the LINGMESS architecture, providing state-of-the-art coreference accuracy, and a substantially faster model, F-COREF, which is the focus of this work. F-COREF allows to process 2.8K OntoNotes documents in 25 seconds on a V100 GPU (compared to 6 minutes for the LINGMESS model, and to 12 minutes of the popular AllenNLP coreference model) with only a modest drop in accuracy. The fast speed is achieved through a combination of distillation of a compact model from the LingMess model, and an efficient batching implementation using a technique we call leftover batching.[1]

## 1 Introduction

Coreference Resolution consists of identifying textual mentions that refer to the same entity in a given text (Karttunen, 1969). This fundamental NLP task can benefit various applications such as Information Extraction (Luan et al., 2018; Li et al., 2020; Jain et al., 2020), Question Answering (Dasigi et al., 2019; Chen and Durrett, 2021), Machine Translation (Stojanovski and Fraser, 2018; Voita et al., 2018), and Summarization (Christensen et al., 2013; Falke et al., 2017; Pasunuru et al., 2021). However, compared to other core tasks such as POS Tagging, named-entity recognition or syntactic parsing, existing packages and state-of-the-art models for coreference resolution are challenging to apply: there are few easy-to-use packages implementing state-of-the-art models, and the available packages consume a lot of GPU memory, and take very long to process each document. For example, the coreference model in the popular AllenNLP package (Gardner et al., 2017), implementing the model of Joshi et al. (2020), requires 27GB of GPU memory and takes 12 minutes to process the 2.8K documents of the OntoNotes corpus, on a V100 GPU.

In this work, we introduce F-COREF, a new open source Python package for simply running an efficient coreference model using a few lines of code. F-COREF predicts coreference clusters 29 times faster than the AllenNLP model (processing the OntoNotes corpus in 25 seconds) and requires only 15% of its GPU memory use, with only a small drop in performance (78.5 vs 79.6 average F1). The package also includes LINGMESS (Otmazgin et al., 2022), a state-of-the-art coreference model, which is almost twice as fast as the AllenNLP model, while being more accurate (81.4 average F1), under the same API.

To achieve F-COREF's speed, we use two additive techniques: model distillation of the strong-but-slow LINGMESS model using large unlabeled data, and an effective batching technique that reduces the number of padded tokens in a batch.

## 2 The F-COREF API

The *fastcoref* Python package is pip installable (`pip install fastcoref`) and provides an easy and fast API for coreference information with only few lines of code without any prepossessing steps.

The F-COREF constructor initializes our pretrained model on a single device:

```
from fastcoref import FCoref

model = FCoref(device='cuda:0')
```

The main functionally of the package is the *predict* function.

```
preds = model.predict(
    texts=['We are so happy to see you
    using our coref package.
    This package is very fast!']
)
```

[1]https://github.com/shon-otmazgin/fastcoref

The return value of the *predict* function is a list of *CorefResult* objects, from which one can extract the coreference clusters (either as strings or as character indices over the original texts), as well as the logits for each corefering entity pair:

```
preds[0].get_clusters()
> [[(0, 2), (33, 36)],
   [(33, 50), (52, 64)]
  ]

preds[0].get_clusters(string=True)
> [['We', 'our'],
   ['our coref package', 'This package']
  ]

preds[0].get_logit(
    span_i=(33, 50), span_j=(52, 64)
)

> 18.852894
```

Processing can be applied to a collection of texts of any length in a batched and parallel fashion:

```
texts = ['text 1', 'text 2',.., 'text n']

# control the batch size
# with max_tokens_in_batch parameter

preds = model.predict(
    texts=texts, max_tokens_in_batch=100
)
```

The `max_tokens_in_batch` parameter can be used to control the speed vs. memory consumption (as well as speed vs. accuracy) tradeoff, and can be tuned to maximize the utilization of the associated hardware.

To use the larger but more accurate LINGMESS model, simply import LingMessCoref instead of FCoref:

```
from fastcoref import LingMessCoref

model = LingMessCoref(device='cuda:0')
```

On top of the provided models, the package also provides the ability to train and distill coreference models on your own data, opening the possibility for fast and accurate coreference models for additional languages and domains.

To summarize, the package provides a simple API that makes predicting coreference entities straightforward and easy-to-use. The package supports any text length as input, and performs efficient batching. The package's F-COREF model is 29 times faster and 4 times smaller than the popular coreference model in the AllenNLP package, while the provided LINGMESS mode is twice as fast the AllenNLP implementation, and more accurate.

## 3 Background: Neural Coreference

Lee et al. (2017) present the first end-to-end model that jointly learns mention detection and coreference decision. Successive follow-up works kept improving performance through the incorporation of widely popular pretrained architectures (Lee et al., 2018; Joshi et al., 2019; Kantor and Globerson, 2019; Joshi et al., 2020). However, as the dimensionality of contextualized encoders increases, keeping in memory all possible span representations becomes highly costly and computationally untractable for long documents.

### 3.1 Faster Neural Coreference

Several methods have been proposed to address this memory constraint at the cost of the computation time and a slight performance deterioration (Xia et al., 2020; Toshniwal et al., 2020; Thirukovalluru et al., 2021). The *s2e* model of Kirstain et al. (2021) managed to improve computation time with a slight *increase* in accuracy.

**s2e** Our F-COREF is based on the architecture of the *s2e* model by Kirstain et al. (2021). Like other neural coreference models, *s2e* scores each pair of spans in the text to be co-referring to each other. However, in order to achieve lower memory footprint s2e moves to representing each span as a function of its start and end tokens. Consequently, the model avoids holding vector representation for each of the $O(n^2)$ spans in memory, and instead stores only $O(n)$ vectors. This reduced memory footprint allows it to handle longer sequences.

The *s2e* architecture includes three components: (1) Longformer (Beltagy et al., 2020), a contextualized encoder; (2) a parameterized *mention scoring function* $f_m$; and (3) a parameterized *pairwise antecedent scoring function* $f_a$. To score any pair of spans to be co-referring, the model starts by encoding the text using Longformer into vectors $x_1, ..., x_n$. Using these vectors, for each possible span $q = (x_k, x_\ell)$ the mention scoring function $f_m(q)$, scores how likely $q$ ("query") being a mention. Then for a pair of spans $c = (x_i, x_j)$, $q = (x_k, x_\ell)$ where $c$ ("candidate") appears before $q$, the pairwise antecedent scoring function, $f_a(c, q)$, scores how likely is $c$ being an antecedent of $q$. In practice, to avoid complexity of $\mathcal{O}(n^4)$, the antecedent function scores only the $\lambda T$ spans with highest mention scores (where $T$ is the number of tokens). Finally, the final pairwise score for a coreference link between $c$ and $q$ is composed by

the score of $q$ being a mention, $c$ being a mention, and how likely is $c$ being an antecedent of $q$:

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

where $\varepsilon$ is the null antecedent.

The computation of $f_m$ and $f_q$ for the entire sequence can be efficiently batched.

**Word-level coreference** Dobrovolskii (2021) proposed moving from scoring pairs of spans to scoring pairs of words, establishing coreference relations between the words, and then expanding each of the relevant words into their mention boundaries. This reduces the model complexity from $\mathcal{O}(n^4)$ to $\mathcal{O}(n^2)$.

### 3.2 What remains slow?

While the *s2e* and the word-level models are considered lightweight and efficient, and substantally improve in speed over Joshi et al. (2019), their computation time is still dominated by their expensive contextualized encoding stage. They also use relatively large hidden layers in their scoring functions (the s2e model has 26 layers and 494M parameters). Thus, one avenue for improving coreference speed is by reducing the model size. Additionally, while batching computations can improve parallelism and thus also throughput, the implementation of batching long documents of varying lengths is often sub-optimal, and results in many padded tokens which translate to wasted computation.

### 3.3 Accurate Neural Coreference

The recent LingMess model of Otmazgin et al. (2022) improves coreference accuracy by observing that different types of entities require different strategies to score, and replacing the single mention-pair scorer with a set of specialized scorers. During inference, each mention pair is deterministically routed to one of the scorers, based on the the type of mentions being scored. This results in state-of-the-art coreference accuracy, while being somewhat less efficient to run and to batch than the *s2e* model.

## 4 Method

We employ two complementary directions in order to obtain a fast and efficient coreference model. First, we substantially reduce the size of the *s2e* model using knowledge distillation (§4.1). Second, our implementation aims to maximize parallelism via batching while limiting the number of unnecessary computations such as padded tokens (§4.2).

### 4.1 Knowledge Distillation

Knowledge Distillation is the process of learning a small student model from a large teacher model.

**Teacher model** We use the state-of-the-art LingMess model of Otmazgin et al. (2022) as the teacher model.

**Student model** we build our student model as a variant of the *s2e* model with fewer layers and parameters. The "expensive" Longformer (Beltagy et al., 2020) encoder was replaced with DistilRoBERTa (Sanh et al., 2019), which is on average $\times 8$ faster than Longformer. The number of parameters of the mention and the antecedent pairwise scorers was reduced by a factor of 6. This reduces the total number of parameters from 494M to 91M. In addition, the number of sequential layers in the network reduced from 26 layers to only 8 layers (6 encoder layers, 1 mention scorer and 1 antecedent scorer). As a result, our student combines the strengths of the *s2e* model by not constructing span representation with a lightweight encoder and substantially less model parameters.

**Hard distillation** Traditional approaches for knowledge distillation trains the student on the logits of the teacher model's predictions on unlabeled data (Gou et al., 2021). However, as we will further elaborate in Section §5.1, applying such an approach to a coreference model with all its components (i.e. encoder, mention scorer, pruning, antecedent scorer) achieves poor performance. To remedy this issue, we employ *hard* target knowledge distillation, where the teacher model acts as an annotator for the unlabeled data and the student model learns from these "silver" annotations.

### 4.2 Maximizing Parallelism and Reducing Unnecessary Computations

**Mention pruning** As mentioned in Section 3, the coreference model computes antecedent scores only for the $\lambda T$ spans with highest mention scores, where $T$ is the number of tokens. As the number of coreferring spans cannot be known in advance, the common approach is to use a soft pruning coefficient ($\lambda = 0.4$) to guarantee high mention recall, while expecting the antecedent scorer to assign

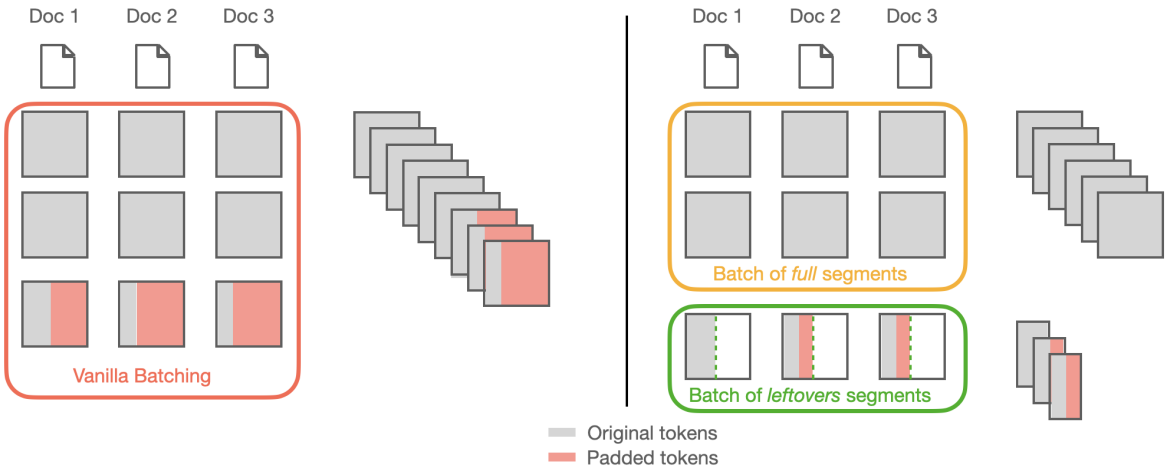


Figure 1: Illustration of document batching using vanilla batching (left) and our leftovers approach (right). In our leftover batching, we create two separate batches, one for the full segments without padding (orange) and one for the leftover segments (green), thus substantially reducing the number of padded tokens (red) compared to the vanilla approach.

a negative score to pairs involving a wrong mention span (Lee et al., 2017; Kirstain et al., 2021). In F-COREF, we adopt a more aggressive pruning ($\lambda = 0.25$), which decreases the number of pairwise comparisons by a factor of 2.56 without harming performance.

**Dynamic batching** We adopt a dynamic batching approach which, given a large number of documents, batches documents until we reach a certain maximum number of tokens. Compared with the naive approach of batching a fixed number of documents together, dynamic batching enables to fully exploit the available memory in our hardware (this approach was also used by Kirstain et al. (2021) when training the *s2e* model).

**Leftovers batching** Figure 1 illustrates our document batching strategy, in comparison with the common approach.

As mentioned in Section 3, the first step in our coreference model consists of encoding the document using a transformer-based encoder. The common approach for encoding long documents with transformer encoders is to split the document into non-overlapping segments of *max_length*, where each segment is encoded separately (Joshi et al., 2019; Xu and Choi, 2020). With that approach, for each long document, we obtain two[2] types of segment lengths: (1) one or more segments of *max_length* i.e. FULL tokens segment, (2) exact one segment $\leq$ *max_length*, i.e the LEFTOVERS tokens segment.

Then to batch multiple documents, a naive but popular approach consists of padding each document's LEFTOVERS segment to *max_length*. This results in a high number of padded tokens, e.g., 34.7% of all tokens are padded when batching 2802 OntoNotes (Pradhan et al., 2012) training set documents with *max_length* = 512. Padded tokens result in unnecessary computations in each layer of the network, as well as unnecessary memory allocation.

To avoid such unnecessary computations, we split each batch into two batches such that the first batch is for the FULL segments and the second batch is for the LEFTOVERS tokens segment of each document. Then we pad the second batch to the max leftovers length rather than padding the leftovers segments to *max_length*. Finally we run the two batches separately and combine them afterwards. With this technique, the padded tokens in the OntoNotes training set reduced dramatically to 0.6%. It should be noted that the aforementioned batching technique is not specific for coreference resolution and can be applied for other tasks that require processing long documents.

## 5 Experiments and Results

**Experiments setup** In our experiments, we use the Multi-news (Fabbri et al., 2019) dataset to train our teacher-student architecture. The Multi-news dataset is an open source dataset aimed at NLP

[2]A document with fewer tokens than *max_length* has only one segment.

| | MUC | | | $B^3$ | | | $CEAF_{\phi_4}$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 | P | R | F1 | Avg. F1 |
| Joshi et al. (2020) | 85.8 | 84.8 | 85.3 | 78.3 | 77.9 | 78.1 | 76.4 | 74.2 | 75.3 | 79.6 |
| Kirstain et al. (2021) | 86.5 | 85.1 | 85.8 | 80.3 | 77.9 | 79.1 | 76.8 | 75.4 | 76.1 | 80.3 |
| Dobrovolskii (2021) | 84.9 | 87.9 | 86.3 | 77.4 | 82.6 | 79.9 | 76.1 | 77.1 | 76.6 | 81.0 |
| Otmazgin et al. (2022) (Teacher) | 88.1 | 85.1 | 86.6 | 82.7 | 78.3 | 80.5 | 78.5 | 76.0 | 77.3 | 81.4 |
| F-COREF OntoNotes only | 78.5 | 84.3 | 81.3 | 68.2 | 74.8 | 71.4 | 64.1 | 72.9 | 68.2 | 73.7 |
| F-COREF Multi-News | 84.8 | 82.8 | 83.4 | 76.8 | 73.7 | 75.2 | 73.8 | 72.7 | 73.2 | 77.4 |
| + FT OntoNotes | 85.0 | 83.9 | 84.4 | 77.6 | 75.5 | 76.6 | 74.7 | 74.3 | 74.5 | 78.5 |

Table 1: Performance on the test set of the English OntoNotes 5.0 dataset. The averaged F1 of MUC, $B^3$, $CEAF_\phi$ is the main evaluation metric.

| | **Runtime** | **Memory** |
|---|---|---|
| Joshi et al. (2020)[1] | 12:06 | 27.4 |
| Otmazgin et al. (2022) (Teacher) | 06:43 | 4.6 |
| + Batching[2] | 06:00 | 6.6 |
| Kirstain et al. (2021) | 04:37 | 4.4 |
| Dobrovolskii (2021) | 03:49 | 3.5 |
| F-COREF | 00:45 | **3.3** |
| + Batching[2] | 00:35 | 4.5 |
| + Leftovers batching[2] | **00:25** | 4.0 |

[1] AllenNLP package implementation.
[2] 10K tokens in a single batch.

Table 2: The inference time(Min:Sec) and memory(GiB) for each model on 2.8K documents. Average of 3 runs. Hardware, NVIDIA Tesla V100 SXM2.

summarization. Each entry in the dataset contains multiple documents and a summary of these documents. For our purposes, we ignored the summaries, and train our student model on the documents, a total of 123,227 documents in the news domain. Furthermore, we use the English portion of the OntoNotes (Pradhan et al., 2012) dataset to evaluate the student model performance (on the test set) and to further fine-tune the student model (on the train set).

The student training procedure includes three phases. In the first phase, we predict coreference clusters on MultiNews using the teacher model (Otmazgin et al., 2022). Secondly, following (Wu et al., 2020; Dobrovolskii, 2021), we pre-train the mention scorer of the student on the output mentions of the teacher, then train the full student model on the predicted teacher coreference clusters. Finally, we finetune the student model on the OntoNotes training set. Implementation details are described in Appendix A.

**Accuracy** Table 1 shows F-COREF's performance on the OntoNotes test set according to the standard evaluation metrics for coreference resolution. F-COREF achieved 78.5 F1 with knowledge distillation and finetuning on OntoNotes. When trained only on OntoNotes, F-COREF achieved only 73.7 F1 (−4.8 F1), showing a substantial benefit from knowledge distillation on the Multi-news dataset (Fabbri et al., 2019). In comparison with other coreference models, F-COREF degrades by 1.1 point compared to the Joshi et al. (2019) model in the AllenNLP package, by 2.9 F1 points compared to LINGMESS (Otmazgin et al., 2022), the teacher, and by 1.8 F1 points versus Kirstain et al. (2021), the *s2e* model, which F-COREF is a variant of. This degradation comes in favor to the model efficiency, which we will discuss in the next paragraph.

**Speed and Memory Usage** Table 2 summarizes the efficiency of the different coreference models. Using the techniques described in Section §4 which reduces the model size, maximize batching and avoids unnecessary computation, the inference time on 2.8K documents is significantly reduced by factor of 9 from the 03:49 minutes of Dobrovolskii (2021)—the fastest model to date—to only 25 seconds for F-COREF.[3]

Most of the speed increase (80%) is due to the smaller model size achieved through distillation, which alone reduces the runtime to 45 seconds. However, introducing batching further reduces the runtime by additional 22% to 35 seconds, and our novel leftover batching reduces further 29%, and gets us to 25 seconds. The more aggressive mention pruning (not shown in the table) had only a negligible additive effect in our experiments, reducing the runtime from 25 to 24 seconds.

Without batching, F-COREF also consumes less

[3] These experiments used batch sizes of 10k tokens. Increasing the batch sizes increase memory consumption, but does not improve overall speed on our NVIDIA Tesla V100 hardware.

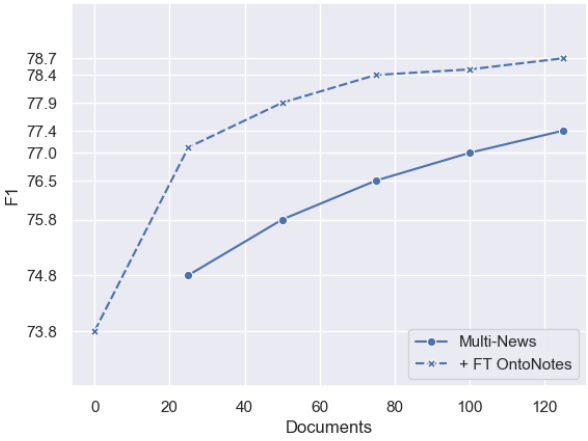


Figure 2: The knowledge distillation learning curve. The x-axis is the number of documents we took from the Multi-News dataset to train the student model. The y-axis is the student F1 score for each size.

memory then Dobrovolskii (2021), a model that recently reduces the coreference complexity from $\mathcal{O}(n^4)$ to $\mathcal{O}(n^2)$.

Finally, compared to one of the most widely used coreference models, the Joshi et al. (2020) model available through the AllenNLP package (Gardner et al., 2017), F-COREF is 29 times faster and consumes 85% less memory.

### 5.1 Further Analysis

**Effect of Unlabeled Data Size** We first analyze the effect of the amount of unlabeled data used in distillation, by training the student model on different amounts of training data. As Figure 2 shows the performance gain between 25K documents and 50K documents is 1 F1 point while the gain between 100K and 125K documents decreases to 0.4 points. This indicates that the gain margin is decreasing, but overall, increasing the dataset size continuously improves the model performance and there is a room for improvement with more data. Fine-tuning the distilled model on the in-domain OntoNotes data consistently improves the results, but is also additive with the distillation: the fine-tuned performance also increases with more unlabaled data in distillation.

**Effect of the Teacher Model** To estimate the effect of the teacher model, we compare the LINGMESS teacher to a *s2e* model (Kirstain et al., 2021) teacher on the same unlabeled data. We obtain 76.6 F1 with *s2e* (vs. 77.4 F1 with LINGMESS) on the OntoNotes (test set) when training only on Multi-News and 78.3 F1 with *s2e* (vs. 78.5 F1 with LINGMESS) after further fine-tuning on OntoNotes (training set). This indicates that the student accuracy increases when more accurate models are used in knowledge distillation, even with hard labels.

**Soft VS. Hard Distillation** Our first attempt to transfer the coreference knowledge from the teacher model to the student model was the traditional knowledge distillation, i.e. soft targets knowledge distillation. For each example in the training set, we forward it first in the teacher model and obtained the top-scoring spans indices, and the pairwise coreference logits. Then we forward the example in the student network (at pruning stage we use the teacher's top-scoring spans indices) to obtain the student coreference pairwise logits. Following common training objective (Hinton et al., 2015; Sanh et al., 2019; Jiao et al., 2020), we optimize the student model using the soft cross entropy loss between the student and the teacher logits.

Our student model reached only 64 F1, while achieving 73.7 F1 without knowledge distillation. Soft distillation presents challenges in coreference models. The main challenge we encounter is at the pruning stage, where both teacher and student should prune the exact same mentions from their individual mention scorer. This forces the student model to learn from a conditional antecedent distribution of the teacher spans indices instead of the full antecedent distribution.

Additionally, we observe that learning to mimic the teacher logits may trouble the training because logits can violate transitivity (e.g positive score for the mention pairs $(a, b)$ and $(b, c)$ but a negative score for $(a, c)$) and propagate contradictory information. Specifically, we build the coreference clusters based on the positive pairwise scores, and verify whether all pairwise scores within the same coreference clusters are positive, as we would naturally expect. In fact, 53.8% of the pairwise scores within the same coreference cluster are negative. In contrast, this undesired behavior does not happen in hard distillation because we assign positive labels for all coreferring antecedents for each mention.

## 6 Conclusions

We introduce the *fastcoref* python package for coreference resolution, and hope its speed an ease of use will facilitate work that utilizes coreference resolution at scale.

## References


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

Jifan Chen and Greg Durrett. 2021. Robust question answering through sub-part alignment. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1251–1263, Online. Association for Computational Linguistics.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2013. Towards coherent multi-document summarization. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1163–1173, Atlanta, Georgia. Association for Computational Linguistics.

Pradeep Dasigi, Nelson F. Liu, Ana Marasović, Noah A. Smith, and Matt Gardner. 2019. Quoref: A reading comprehension dataset with questions requiring coreferential reasoning. 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 5925–5932, Hong Kong, China. Association for Computational Linguistics.

Vladimir Dobrovolskii. 2021. Word-level coreference resolution. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7670–7675, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Alexander Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. 2019. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1074–1084, Florence, Italy. Association for Computational Linguistics.

Tobias Falke, Christian M. Meyer, and Iryna Gurevych. 2017. Concept-map-based multi-document summarization using concept coreference resolution and global importance optimization. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 801–811, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

Jianping Gou, B. Yu, Stephen J. Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *ArXiv*, abs/2006.05525.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531.

Sarthak Jain, Madeleine van Zuylen, Hannaneh Hajishirzi, and Iz Beltagy. 2020. SciREX: A challenge dataset for document-level information extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7506–7516, Online. Association for Computational Linguistics.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

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

Mandar Joshi, Omer Levy, Luke Zettlemoyer, and Daniel Weld. 2019. BERT for coreference resolution: Baselines and analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5803–5808, Hong Kong, China. Association for Computational Linguistics.

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.

Lauri Karttunen. 1969. Discourse referents. In *International Conference on Computational Linguistics COLING 1969: Preprint No. 70*, Sånga Säby, Sweden.

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.

Manling Li, Alireza Zareian, Ying Lin, Xiaoman Pan, Spencer Whitehead, Brian Chen, Bo Wu, Heng Ji, Shih-Fu Chang, Clare Voss, Daniel Napierski, and Marjorie Freedman. 2020. GAIA: A fine-grained multimedia knowledge extraction system. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 77–86, Online. Association for Computational Linguistics.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232, Brussels, Belgium. Association for Computational Linguistics.

Shon Otmazgin, Arie Cattan, and Yoav Goldberg. 2022. Lingmess: Linguistically informed multi expert scorers for coreference resolution. *ArXiv*, abs/2205.12644.

Ramakanth Pasunuru, Mengwen Liu, Mohit Bansal, Sujith Ravi, and Markus Dreyer. 2021. Efficiently summarizing text and graph encodings of multi-document clusters. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4768–4779, Online. 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.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

Dario Stojanovski and Alexander Fraser. 2018. Coreference and coherence in neural machine translation: A study using oracle experiments. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 49–60, Brussels, Belgium. Association for Computational Linguistics.

Raghuveer Thirukovalluru, Nicholas Monath, Kumar Shridhar, Manzil Zaheer, Mrinmaya Sachan, and Andrew McCallum. 2021. Scaling within document coreference to long texts. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3921–3931, Online. Association for Computational Linguistics.

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

Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. 2018. Context-aware neural machine translation learns anaphora resolution. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1264–1274, Melbourne, Australia. Association for Computational Linguistics.

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

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.

## A Implementation details

As mentioned in the paper (Section 5), we train our student model on the teacher predictions that we obtained on the source documents of the Multi-News dataset. Due to LINGMESS limitations, we omit source documents with more than 4096 tokens from the original dataset. We tokenize each document with Spacy and run the teacher model to predict coreference clusters.

We implement our model using Pytorch, HuggingFace and the Datasets library. We pre-train the mention scorer of our model on the teacher predicted mentions by optimizing the binary cross-entropy loss for each mentions. We use a learning rate of $3e-04$ for the mention scoring function and $1e-05$ for the distillRoBERTa encoder. Then, we train the full student model on the predicted clusters with $\lambda = 0.25$ using the marginal log likelihood loss, as common in coreference resolution. At this stage, we use a learning rate of $3e-04$ for the antecedent and scorer and a smaller learning rate, $1e-05$, for the mention scorer and encoder in order to finetune them. We further fine-tune our full student model on OntoNotes using a learning rate of $1e-05$ for all parameters.