Scaling Within Document Coreference to Long Texts

Raghuveer Thirukovalluru[1], **Nicholas Monath**[1], **Kumar Shridhar**[2], **Manzil Zaheer**[3], **Mrinmaya Sachan**[2]**and Andrew McCallum**[1]
[1]UMass Amherst, [2]ETH Zurich, [3]Google
rthirukovall@umass.edu, {nmonath,mccallum}@cs.umass.edu
{shridhar.kumar,mrinmaya.sachan}@inf.ethz.ch
manzilzaheer@google.com

Abstract

State of the art end-to-end coreference resolution models use expensive span representations and antecedent prediction mechanisms. These approaches are expensive both in terms of their memory requirements as well as compute time, and are particularly ill-suited for long documents. In this paper, we propose an approximation to end-to-end models which scales gracefully to documents of any length. Replacing span representations with token representations, we reduce the time/memory complexity via token windows and nearest neighbor sparsification methods for more efficient antecedent prediction. We show our approach's resulting reduction of training and inference time compared to state-of-the-art methods with only a minimal loss in accuracy.

1 Introduction

Recent advances in coreference resolution (Lee et al., 2018; Joshi et al., 2019, 2020; Wu et al., 2020) have been largely based on the end-to-end model proposed by Lee et al. (2017). However, these models are costly both in terms of training and inference time, as well as memory requirements, especially for long documents. The large computational cost makes the models infeasible to run for a typical user on large document collections in domains such as blogs, stories, books, etc. Moreover, a reduction in energy use of these models can be of benefit to cloud service providers' costs and there can also be environmental benefits (Strubell et al., 2019; Schwartz et al., 2020).

There are two main computational bottlenecks in using end-to-end coreference models on long documents: (i) span and span-pair representations for all spans in the document are simultaneously considered, and (ii) the coreference decision for a mention requires considering all candidate antecedent spans.

In this paper, we propose an approximation to the end-to-end coreference model (Lee et al., 2017) that scales to long documents by addressing both these bottlenecks. Our proposed approach operates at the token level instead of the span level, removing the quadratic dependence on the number of mention spans in a document and addressing bottleneck (i). We propose token level scoring functions for the bilinear inference model originally proposed by Lee et al. (2018). To address bottleneck (ii), we use token windows, token-level k-nearest neighbor relationships along with low-rank matrix approximations of the token similarity matrix thereby improving time/memory efficiency. We also propose an approach to drop token representations from memory to reduce memory requirements while still maintaining the accuracy.

We evaluate our approach on three coreference datasets: CoNLL-2012 (Pradhan et al., 2012), Litbank (Bamman et al., 2019), and MedMentions (Mohan and Li, 2019) and observe competitive accuracy to state of the art coreference models based on end-to-end training while achieving both faster training and inference running times. Our approach is also more memory efficient and up to 10x faster than the recently proposed memory-based incremental coreference resolution model on Litbank (Toshniwal et al., 2020b). Finally, we demonstrate the scalability of our approach by running it on a novel of two million tokens in 14 minutes while requiring just 12GB of GPU RAM, while previous work can only scale to documents of just around eleven thousand tokens even with up to 48GB of GPU RAM.

Concurrent to our work, Kirstain et al. (2021) also proposes a bilinear token level scoring function for coreference. The focus of our work however

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3921–3931

is on long documents and we further introduce a token k-nn graph approximation, a low-rank matrix factorization and an approach to drop non essential candidate antecedents to improve mem/time scalablity.

2 Background: End-to-end Within-Document Coreference

End-to-end within-document coreference resolution models jointly discover a set of mentions, $\mathcal{M}$ in a document $\mathcal{D}$ and determine which of the mentions are coreferent. We use $\mathcal{D}$ to refer to the ordered set of tokens in the document $\mathcal{D} = \{x_1, x_2, \ldots, x_T\}$. Each mention is a token span $s = x_i, \ldots, x_j$[1]. We use $\boldsymbol{x}_i$ to refer to the contextualized embedding of the token i (see Section 4.1 for more details on the encoder). The model comprises of two parts which are jointly trained: (a) a mention-proposer, and (b) an antecedent-predictor. The mention-proposer model evaluates all spans $\mathcal{S}$ in the dataset and proposes a small set of potential mentions $\mathcal{M} \subset \mathcal{S}$. The antecedent prediction model evaluates the mentions suggested by the mention proposer and produces coreference clusters (chains) $\mathcal{C} \subset \mathcal{P}(\mathcal{M})$, where $\mathcal{P}(\cdot)$ is the powerset.

Recent work (Lee et al., 2018; Joshi et al., 2020; Xu and Choi, 2020, inter alia) has built upon the first neural, end-to-end coreference model (Lee et al., 2017). Each of these models introduce two scoring functions $\mathsf{s_m}(s)$ and $\mathsf{s_a}(m_1, m_2)$. $\mathsf{s_m}(s)$ represents the scores that a span s is a mention, and $\mathsf{s_a}(m_1, m_2)$ is the score for mention m_1 being an antecedent of mention m_2. These scoring functions are used to define the joint mention proposal and antecedent prediction model for coreference.

Mention proposer: The previous works use a neural network for $\mathsf{s_m} : \mathcal{S} \rightarrow \mathbb{R}$. The architecture takes in a mention span and outputs a score. For each mention span s, the model computes a vector representation $\boldsymbol{g}_s \in \mathbb{R}^d$. The scoring functions take these vector representations as input:

$$\mathsf{s_m}(s) = \boldsymbol{w}_\text{m} \cdot \text{FFNN}_\text{m}(\boldsymbol{g}_s) \quad (1)$$

and $\boldsymbol{g}_s$ is computed as:

$$\boldsymbol{g}_s = [\boldsymbol{x}_{\text{START}(s)}, \boldsymbol{x}_{\text{END}(s)}, \hat{\boldsymbol{x}}_s, \phi(s)] \quad (2)$$

[1]In our work as well as most work on within document coreference (Lee et al., 2017, 2018; Joshi et al., 2019) we only consider contiguous mention spans rather than allowing spans to be arbitrary sets of tokens (non-contiguous).

where $\boldsymbol{x}_{\text{START}(s)}, \boldsymbol{x}_{\text{END}(s)}$ are the boundary representations of span s, $\hat{\boldsymbol{x}}_s$ is a self-attention representation of span s, and $\phi(s)$ encodes the width (number of tokens) of span s. For efficiency, the model selects top $0.4T$ scoring mention spans where T is the number of tokens of the document. We refer to this set of selected mention spans as $\mathcal{M}$. We use an ordering of the mentions $m \in \mathcal{M}$ based on their start/end offsets.

Antecedent Prediction: Previous work has explored several models for antecedent prediction. The most computationally efficient being a bilinear scoring model (Lee et al., 2018):

$$\mathsf{s}^{\text{bi}}_\mathsf{a}(m_1, m_2) = \mathbf{g}_{s_{m_1}} W^T \mathbf{g}_{s_{m_2}} \quad (3)$$

Higher-order inference models, which use deep models to capture coreference relationships between mentions, have also been considered (Lee et al., 2018). For example,

$$\mathsf{s}^{\text{hoi}}_\mathsf{a}(m_1, m_2) = \boldsymbol{w}_\text{a} \cdot \text{FFNN}_\text{a}([\boldsymbol{g}_{m_1}; \boldsymbol{g}_{m_2}; \boldsymbol{g}_{m_1} \odot \boldsymbol{g}_{m_2}, \phi_{m_1,m_2}]) \quad (4)$$

We refer the reader to Xu and Choi (2020) for a detailed analysis of higher order inference models.

The prediction of the antecedent of each mention, which we refer to as *inference*, is done by *backwards chaining*. Clusters of mentions are determined by finding for each mention, the highest scoring antecedent among the mentions appearing earlier in the document and adding the mention to the antecedent's cluster. This can be described as finding the connected components of a graph G. Coarse-to-fine inference (Lee et al., 2018) and the standard bi-linear model can be differentiated by different ways of constructing the adjacency matrix of the graph G with nodes being the mentions $\mathcal{M}$. We refer to this adjacency matrix as A, and use $A_{i,j} = 1$ to indicate the existence of an edge between mention m_i and m_j. The adjacency matrix of the bilinear model can be written as:

$$A^{\text{bi}}_{i,j} = \mathbb{I}[j = \underset{h \leq i}{\text{argmax}}\, \mathsf{s}^{\text{bi}}(m_i, m_h)] \quad (5)$$

$$\mathsf{s}^{\text{bi}}(m_i, m_j) = \mathsf{s}^{\text{bi}}_\mathsf{a}(m_i, m_j) + \mathsf{s_m}(m_i) + \mathsf{s_m}(m_j) \quad (6)$$

The adjacency matrix of the higher-order model can be written as:

$$A^{\text{hoi}}_{i,j} = \mathbb{I}[j = \underset{h \leq i}{\text{argmax}}\, \mathsf{s}^{\text{hoi}}(m_i, m_h)] \times \mathbb{I}[j \in \underset{h \leq i}{\text{argtopk}}\, \mathsf{s}^{\text{bi}}(m_i, m_h)] \quad (7)$$

$$\mathsf{s}^{\text{hoi}}(m_i, m_j) = \mathsf{s}_{\mathsf{a}}^{\text{hoi}}(m_i, m_j) + \mathsf{s}_{\mathsf{a}}^{\text{bi}}(m_i, m_j) + \mathsf{s}_{\mathsf{m}}(m_i) + \mathsf{s}_{\mathsf{m}}(m_j) \quad (8)$$

where k for argtopk is a hyperparameter.

End-to-end training The mention proposal and antecedent prediction models are trained by relaxing the adjacency matrix A, replacing the argmax operation with a softmax (i.e., setting a weighted edge between i and j with weight $\mathsf{s}(m_i, m_j)$). The training objective is to maximize the log-likelihood of a ground truth adjacency matrix $A^\star$, where $A^\star_{i,j} = 1$ if m_i and m_j are coreferent and $i < j$ under the relaxed adjacency matrix. The argtopk operation is not relaxed. A *nil* antecedent is introduced, which provides similarity (s_a) of 0 to any mention span is incorporated in the training objective. The number of candidate antecedents is also restricted by a hyperparameter (Lee et al., 2018).

3 Efficient Approximations for End-to-End Coreference

We describe our proposed approach for efficiently approximating the *span-based* end-to-end coreference model with a *token-level* model. Our model jointly predicts which tokens are in the same mention spans (i.e., mention proposal) and what tokens are coreferent with one another (i.e., antecedent prediction). By operating at the token level, we remove the dependence on considering quadratically many phrases. We show the structure of our approximation allows for a *sparsification* technique that reduces the number of antecedent predictions that need to be considered using k-nearest neighbor relationships between tokens and by splitting documents into windows with certain computations made independently for each window. We describe how low-rank matrix approximations can be used to improve inference efficiency.

3.1 Mention Proposer

Observe that computing the set $\mathcal{M}$ requires us to evaluate $\mathsf{s}_\mathsf{m}(\cdot)$ for all candidate spans $\mathcal{S}$ in the document (which grows roughly quadratically with the number of tokens). Recall that $\mathsf{s}_\mathsf{m}(\cdot)$ is a function of the start and end tokens of each span, producing a score that is high if the pair of tokens likely form a span. This approach can be thought of as having each token t in the document predict whether or not another token u is the last token in a span beginning with t.

We first model for each token t whether it is a start (st) or end (en) token of some phrase using a linear model:

$$M_t^{\mathsf{st}} = w_{\mathsf{st}}^T \boldsymbol{x}_t \qquad M_t^{\mathsf{en}} = w_{\mathsf{en}}^T \boldsymbol{x}_t \quad (9)$$

These terms weighs each token by how likely it is to be part of *some* mention span.

Following Kirstain et al. (2021), we find that there can an empirical benefit(described in Section 6) to additionally modelling the relationship between u and t, i.e., whether it is reasonable for the span beginning with t to end in u. To do this we use a asymmetric (bilinear) scoring function. Further, we restrict the spans to be contiguous and follow the rule-based span criteria of previous work (Lee et al., 2017).

$$M_{t,u} = \begin{cases} \boldsymbol{x}_t^T W_{\mathsf{M}} \boldsymbol{x}_u & t \text{ to } u \text{ is a valid span} \\ -\infty & \text{otherwise} \end{cases} \quad (10)$$

For each token, we predict candidate end-tokens for a mention span starting at the given token. We assign a score per span by sum of Eq. 9 & 10 and follow previous work to select the top $0.4T$ scoring spans (mentions). We can replace the mention scoring mechanism used in previous works $\mathsf{s}_\mathsf{m}(\cdot)$ with an approximation based on the token level score:

$$\widehat{\mathsf{s}_\mathsf{m}}(m_i) = M^{\mathsf{st}}_{\text{START}(m_i)} + M^{\mathsf{en}}_{\text{END}(m_i)} + M_{\text{START}(m_i),\text{END}(m_i)} \quad (11)$$

Rather than having to instantiate a d-dimensional span representation for all $|\mathcal{S}|$ spans, our approach simply uses the output token representations from the encoder. This requires $O(T)$ space compared to $O(|\mathcal{S}|)$ space. Note that computing $\widehat{\mathsf{s}_\mathsf{m}}(m_i)$ for all mentions requires at most two matrix multiplications, each with just T rows. Observe that this leads to a drastic reduction in time and space complexity. As noted by previous work (Toshniwal et al., 2020b), the mention proposal step requires the most memory usage because of the quadratic dependency. We validate the reduction in time and memory requirements of our token level mention detection in Section 4.5 & 4.6.

Pretraining For Mention Detection Previous work (Wu et al., 2020; Toshniwal et al., 2020b) has shown that pre-training models for mention detection is beneficial, especially in cases where

predicting singleton mentions is required (e.g., LitBank (Bamman et al., 2019)). Given a set of ground truth mentions $\mathcal{M}^\star$ and the set of mentions from a given document $\mathcal{M}$, we use a mention detection loss which minimizes:

$$-\sum_{m_i \in \mathcal{M}} \log \sigma(\widehat{\mathsf{s_m}}(m_i)) \mathbb{I}[m_i \in \mathcal{M}^\star] + \log \sigma(1 - \widehat{\mathsf{s_m}}(m_i)) \mathbb{I}[m_i \notin \mathcal{M}^\star] \quad (12)$$

We use it as a multi-task objective in training the models and as well as a pre-training objective. We detect singleton mentions by using a threshold on the mention score value $\widehat{\mathsf{s_m}}(m_i)$ which is tuned on the development set according to the downstream performance.

3.2 Antecedent Scoring

Next, we would like to model coreference relationships between tokens to approximate the span-level scoring function ($\mathsf{s_a}$, Eq. 3, 5). We predict for each token, the other tokens with which it is coreferent. These predictions are then aggregated to make span level predictions.

First, we consider approximating bilinear scoring function at the token level ($\mathsf{s}^{\mathsf{bi}}_{\mathsf{a}}$). We use a bilinear model applied to the encoded token representations. We parameterize four asymmetric similarity functions. Note that the backwards-chaining property of inference motivates our use of the asymmetric function. We model the similarity between tokens that are the start or end tokens of phrases separately:

$$S^{\mathsf{ss}}_{i,j} = \boldsymbol{x}_{\text{START}(i)} W^T_{\mathsf{st}} \boldsymbol{x}_{\text{START}(j)} \quad (13)$$

$$S^{\mathsf{es}}_{i,j} = \boldsymbol{x}_{\text{END}(i)} W^T_{\mathsf{st}} \boldsymbol{x}_{\text{START}(j)} \quad (14)$$

$$S^{\mathsf{se}}_{i,j} = \boldsymbol{x}_{\text{START}(i)} W^T_{\mathsf{en}} \boldsymbol{x}_{\text{END}(j)} \quad (15)$$

$$S^{\mathsf{ee}}_{i,j} = \boldsymbol{x}_{\text{END}(i)} W^T_{\mathsf{en}} \boldsymbol{x}_{\text{END}(j)} \quad (16)$$

We use these similarities to approximate the bilinear antecedent scoring function (Eq. 3) as:

$$\widehat{\mathsf{s}^{\mathsf{bi}}_{\mathsf{a}}}(m_i, m_j) = S^{\mathsf{ss}}_{i,j} + S^{\mathsf{es}}_{i,j} + S^{\mathsf{se}}_{i,j} + S^{\mathsf{ee}}_{i,j} \quad (17)$$

Observe how $\widehat{\mathsf{s}^{\mathsf{bi}}_{\mathsf{a}}}(\cdot,\cdot)$ reduces the memory requirements compared to $\mathsf{s}^{\mathsf{bi}}_{\mathsf{a}}(\cdot,\cdot)$. We do not need to instantiate the span representations, only use the encoded token representations. Computing $\widehat{\mathsf{s}^{\mathsf{bi}}}(m_i, m_j)$ requires at most $O(T^2)$ instead of $O(|\mathcal{S}|^2)$ work. We can compute each of the $S^{\cdot}_{i,j}$ (Eq. 16) in space $O(T^2)$ and as matrix multiplication between matrices of $O(T)$ rows.

For training and inference in our model, we define the adjacency matrix A with $\hat{A}$:

$$\widehat{A^{\mathsf{bi}}}_{i,j} = \mathbb{I}[j = \operatorname*{argmax}_{h \le i} \widehat{\mathsf{s}^{\mathsf{bi}}}(m_i, m_h)] \quad (18)$$

$$\widehat{\mathsf{s}^{\mathsf{bi}}}(m_i, m_j) = \widehat{\mathsf{s}^{\mathsf{bi}}_{\mathsf{a}}}(m_i, m_j) + \widehat{\mathsf{s_m}}(m_i) + \widehat{\mathsf{s_m}}(m_j) \quad (19)$$

Inference can then be done as exactly as before, using connected-components based inference.

3.3 Token Windows & Sparsifying Antecedent Scoring with k-NN Graphs

We can use the backwards-chaining structure of the inference procedure and divide a document into smaller token windows (non-overlapping), reducing the number of tokens that need to be encoded in any one component. We can propose mentions independently in each window. We then perform antecedent scoring using the K-NN sparsification described below for each window. By batching the long document into these windows, we never need to store more than the final encoded token representations for the tokens appearing in some entity cluster.

The approximation method presented thus far reduces the complexity of end-to-end coreference approaches from depending on the number of spans to the number of tokens. However, for long documents scaling quadratically in the number of tokens is still prohibitively expensive, both in terms of time complexity and also in terms of memory. Observe that computing and storing each of the $\widehat{\mathsf{s}^{\mathsf{bi}}}(\cdot,\cdot)$ may become prohibitively expensive for all pairs of tokens in the document. We would like to reduce the time and space complexity of this approach.

We propose to approximate the top scoring pairs of mention spans according to $\widehat{\mathsf{s}^{\mathsf{bi}}}(\cdot,\cdot)$ (i.e., further approximating $\mathsf{s}^{\mathsf{bi}}(\cdot,\cdot)$). We do this by only allowing two mentions m_i and m_j to be coreferent if the start/end tokens of m_j are in the k-nearest neighbors of the start/end tokens of m_i. More precisely, we will maintain the k nearest neighbors of each token for each of the four similarity functions S^{ss}, S^{se}, S^{es}, S^{ee} (Eq. 16). To align with inference

procedure, we select these k nearest neighbors for each token only from the preceding tokens in the document. we define $S^{\mathsf{ss}}_{\text{knn}i,j}$ to be non-zero only if j is in the top-k values of $S^{\mathsf{ss}}_{i,\cdot}$

$$S^{\mathsf{ss}}_{\text{knn}i,j} = \begin{cases} S^{\mathsf{ss}}_{i,j} & \text{if } j \in \text{argtopk}_h\, S^{\mathsf{ss}}_{i,h} \\ 0 & \text{otherwise} \end{cases} \quad (20)$$

We define $S^{\mathsf{se}}_{\text{knn}i,j}, S^{\mathsf{es}}_{\text{knn}i,j}, S^{\mathsf{ee}}_{\text{knn}i,j}$ analogously.We then build an further approximation of $\widehat{\mathsf{s}^{\mathsf{bi}}}$ using these $S^{\cdot}_{\text{knn}}$ values:

$$\widehat{\mathsf{s}^{\mathsf{bi}}_{\mathsf{a}\,\text{knn}}}(m_i, m_j) = S^{\mathsf{ss}}_{\text{knn}i,j} + S^{\mathsf{es}}_{\text{knn}i,j} + S^{\mathsf{se}}_{\text{knn}i,j} + S^{\mathsf{ee}}_{\text{knn}i,j} \quad (21)$$

Observe that storing $S^{\mathsf{ss}}_{\text{knn}i,j}$ can use sparse matrices and therefore provide better scalability to long documents for which storing $O(4Tk)$ is advantageous over $O(T^2)$.

End-to-end Training We use the same end-to-end training procedure that was used by previous work (Section 2) using our approximated mention proposal and antecedent scoring procedures. We note that the use of token windows and KNN sparsification of the antecedent scoring term do not change training at all, this is only applied at inference time.

3.4 Low-dimensional Approximations

Much of the computation time of the k-NN graph approximation model comes from the computation of the top-k nearest tokens. The computation bottleneck mostly depends on the high dimensionality of the encoded token representations, which are from transformer-based language models (Joshi et al., 2020).

To produce lower dimensional embeddings of each token, which preserve similarities in the original space, we use low-rank matrix approximation methods, specifically the Nyström method (Williams and Seeger, 2001; Musco and Musco, 2017, inter alia). We hope to approximate the matrices $S^{\mathsf{ss}}_{i,j}, S^{\mathsf{se}}_{i,j}, S^{\mathsf{es}}_{i,j}, S^{\mathsf{ee}}_{i,j}$. While these are asymmetric, we can consider an equivalent symmetrized version where each token appears two times (on left/right of bilinear term) to apply Nyström. The lower dimensional embedding produced by Nyström is done

The Nyström method provides a low-rank approximation of a symmetric pairwise similarity matrix $S \in RR^{N\times N}$, by selecting ℓ *landmark* points uniformly at random among the N rows of S. We use L_i to be column vector one-hot representation of the i^{th} landmark. We assume $L \in \mathbb{R}^{N\times\ell}$ to be a matrix of such one-hot representations. The approximation of S is given by: $\hat{S} = LS(L^TSL)^{-1}L^TS$. The term LS is a ℓ dimensional embedding of the rows, which is defined by the similarity of each row with each of the ℓ landmarks (ℓ is the reduced dimension). Similarly, $(L^TSL)^{-1}L^TS$ can be thought of as providing a ℓ dimensional embedding of each column of S, which is based on the similarity and the (inverse) of the landmark similarities.

3.5 Limiting Num. of Candidate Antecedents

In the aforementioned approach, the number of candidate antecedents scales with the document length. We would like to determine a mechanism for using a fixed number of candidate antecedents if desired. Previous work other work uses entity-level representations to achieve this (Toshniwal et al., 2020b; Xia et al., 2020).

In our work, we operate at the mention level, removing mentions as candidate antecedents. We define a hyperparameter, ρ, which is maximum number of antecedents that would be kept after processing each window of the document. Our approach removes mentions as candidate antecedents which (1) belong to large coreference clusters (2) are not frequently selected as antecedents. We achieve this by dropping mentions in the order of $|\mathcal{C}_m|$ - $\sum_i A_{i,m}$, where $|\mathcal{C}_m|$ is the size of the cluster of the mention m and $\sum_i A_{i,m}$ is the degree of the mention in the antecedent graph.

4 Experiments

In this section, we compare our proposed approach for scalable coreference on long documents to various state-of-the-art methods in terms of accuracy as well as efficiency of training and inference. We perform a detailed scalability analysis, which characterizes the time/memory used by each method as a function of the length of documents. We also report timing results on novels of $\sim$ 2 million tokens.

4.1 Datasets

We evaluate each method on the following datasets: **CoNLL-2012 Shared Task:** The CoNLL-2012

shared task (Pradhan et al., 2012) uses the v5.0 of the OntoNotes corpus for the task of coreference resolution in English, Chinese, and Arabic languages. We use only the English version for our experiments. The training set contains 2802 training, 343 development, and 348 test documents. The training documents contain on average of 454 tokens and a maximum of 4009 tokens. **Litbank:** We also use the Litbank dataset (Bamman et al., 2019) which consists of 210,532 tokens evenly drawn from 100 different English language literary texts. The average document length in Litbank is much longer (around 2,000 tokens). Following (Bamman et al., 2019; Toshniwal et al., 2020b), we use a 10-fold cross-validation setup with 80% of the data as training data and rest 10% each as validation and test data. The final evaluation is reported as the average of all 10 test runs. Note that the family of end-to-end approaches that we are approximating with our method do not predict singletons as is typically done for Litbank. Mention pretraining is performed as described in Section 3.1 **MedMentions** We also repurpose MedMentions (Mohan and Li, 2019) an existing entity linking dataset in the biomedical domain for coreference resolution. We treat the entity labels as the ground truth cluster assignments of each mention for coreference training/analysis. We use the ST21PV subset that is recommended by (Mohan and Li, 2019). **Artamène ou le Grand Cyrus (Artamène, or Cyrus the Great)**. To further asses the scalability of our approach, we run our method on an English translation of the 17th century French novel that is one of the longest books available in English the public domain (Scudery, 1601). The work contains *1.99 million* tokens and over two million sub-tokens. We use this data to illustrate the scalability of our approach to really long documents.

4.2 Methods

We compare our end-to-end coreference approximation with and without the token windows, KNN sparsification approach (i.e. Ours and Ours (Sp.), Section 3.3). We denote the number of neighbors used in the sparsification approach as k and the size of the window used as w. We compare these to the methods that they are approximating: the bilinear scoring function-based method (E2E (bi)) (Lee et al., 2017) as described in Eq. 3 and the coarse-to-fine higher-order inference based approach (E2E (hoi)) (Lee et al., 2018). All models use spanbert-large (Joshi et al., 2020) to encode tokens. The encoder parameters are trained along with the coreference specific model parameters (see Section 4.4 for details). E2E (bi), E2E (hoi) use additional features such as speaker and genre, we do not use this metadata in our proposed approximation approach.

4.3 Coreference Performance

In Table 1, we report the coreference performance (along with the running time and memory usage) for each method on the three datasets. We observe that our approximate approach is achieves comparable performance to the E2E approaches on CoNLL-2012 and MedMentions, performing slightly worse on Litbank. We hypothesize that the token level representations can be effective at these tasks due to the expressiveness of the contextualized embeddings. We observe that the performance of our model is relatively unchanged with and without the sparsification approach applied.

Recently, Toshniwal et al. (2020b); Xia et al. (2020) have proposed memory-based models optimising memory usage. Toshniwal et al. (2020b) trains for improve mention detection by another pretraining process. These papers achieve state-of-the-art results on Litbank and are focused on reducing the running time and memory usage of coreference models by storing *entity* representations instead of *mention* representations in a bounded memory architecture. We compare inference running time and coreference performance of our method with them in Table 1. We find that our models run 10x faster and are slightly more memory efficient than UMem(Toshniwal et al., 2020b) while matching their performance on litbank.

4.4 Experimental Details

We use the hyperparameter settings from (Xu and Choi, 2020) in all applicable cases. We use 512 as the segment length. On CoNLL and Medmentions we train all models for 24 epochs with maximum training sentences set to 3. On Litbank, we train for 120 epochs and pick parameters from (Toshniwal et al., 2020b). We use 0.4 as the ratio to pick the top spans(mentions) among all candidate spans.

4.5 Inference Time and Memory Usage

In Figure 1, we compare the time and memory used by the end-to-end coreference models and our

		Inference		Overall	MUC			CEAF			B^3		
		Mem.	Time	F1	R	P	F1	R	P	F1	R	P	F1
CoNLL	E2E (hoi)	7.77	30	**79.6**	84.8	86.1	85.4	77.3	79.3	78.3	74.7	76	75.4
	E2E (bi)	4.31	28.2	78.4	85.03	84.1	84.56	76.7	76.56	76.63	75.98	72.44	74.17
	Ours	1.78	**24.79**	78.03	84.22	84.28	84.25	74.43	72.63	73.52	75.93	76.68	76.3
	Ours (Sp.)	**1.5**	26.81	77.59	83.34	84.43	83.88	74.39	72.13	73.24	74.36	76.99	75.65
Litbank	E2E (hoi)	2.78	8.12	**78.44**	91.92	87.24	89.50	66.47	68.90	67.59	80.33	76.28	78.21
	E2E (bi)	2.68	5.2	77.77	91.72	86.73	89.15	64.82	68.72	66.67	78.74	76.33	77.48
	UMem	3.00	23.46	76.5	85.7	90.8	88.2	66.0	65.1	65.5	72.1	80.0	75.9
	Ours	**2.12**	2.1	75.93	89.53	86.28	87.86	65.09	65.38	65.18	73.65	76.00	74.75
	Ours (Sp.)	2.39	**1.57**	74.71	87.90	86.14	86.99	65.19	63.67	64.36	70.82	74.95	72.77
MedMent.	E2E (hoi)	3.9	57.08	60.86	63.26	65.75	64.48	57.6	61.29	59.38	60.1	57.37	58.71
	E2E (bi)	2.64	54.48	60.67	64.69	64.5	64.6	59.21	59.53	59.37	61.05	55.32	58.04
	Ours	1.69	**43.9**	**61.76**	67.92	63.02	65.38	59.78	59.55	59.67	63.60	57.17	60.21
	Ours (Sp.)	**1.57**	45.3	61.49	66.87	63.48	65.13	59.65	59.34	59.49	62.12	57.69	59.82

Table 1: **Accuracy & Efficiency** on the three benchmarks. Memory in GB, time in seconds.

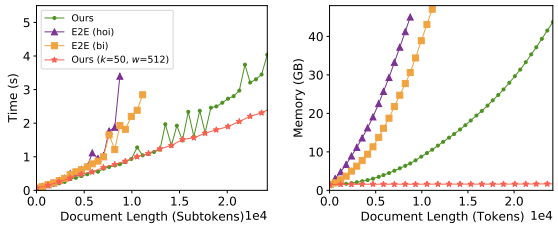

Figure 1: **Time and GPU Memory Comparison of different models on the book "Little Women".**

		Training	
		Time	Mem.
CoNLL	Ours	**6.8**	**13.69**
	E2E (hoi)	11.75	19.39
	E2E (bi)	7.5	15.7
Litbank	Ours	**2.9**	**19.43**
	E2E (bi)	3.2	22.7
	E2E (hoi)	5	27.0
MedM.	Ours	**5.5**	**13.8**
	E2E (hoi)	8.75	18.5
	E2E (bi)	6.25	15.6

Table 2: **Training time(hours) and memory(GB) usage**. Our approach requires less time and memory than the competing end-to-end approaches.

proposed family of approximations. We select a book at random from the Litbank corpus (*Little Women*) and report the time and memory used by each method to perform coreference as a function of the number tokens analyzed. We plot a curve for each, reporting the statistics until the method runs out of GPU memory (48GB). We cut off the x-axis of the graph where our proposed approach without the backwards chaining runs out of memory. Our token level models only scales upto 24K tokens. We note that Ours (Sp.) is able to run on the entire book requiring only marginally higher memory for higher document lengths. This is in contrast with previous E2E methods which run out of memory for documents longer than 1e4 tokens.

4.6 Training Time and Memory Usage

We report in Table 2 the training time and memory requirements for each of the methods. For each dataset, we train all the methods in focus for the same number of epochs/updates. We train for 24 epochs on CoNLL, 120 epochs on Litbank and 24 epochs on MedMentions. We observe that our approach greatly reduces GPU memory requirements and are also slightly faster. This gap is wider for datasets containing longer documents as shown by the numbers on LitBank. Note that the sparsification approximation is simply an inference time approximation and uses the same trained model as our approach with the K-NN approximation.

4.7 Scaling to Long Documents

We run Ours (Sp.) on the full text of *Artamène or Cyrus the Great*, which has 1.99M tokens ($>$ 2M subtokens). To our knowledge, this is the largest single document a neural within document coreference system has been applied to. In Figure 2, we show that our approach runs in about 14 minutes. Further, we demonstrate how the hyperprameters of the sparsification can be adjusted depending on the system requirements. We show that the window size parameter can be set to be the minimal amount (w=512) to require just 13 GB of GPU RAM. Table 3 suggests using small window sizes are also advantageous in terms of accuracy.

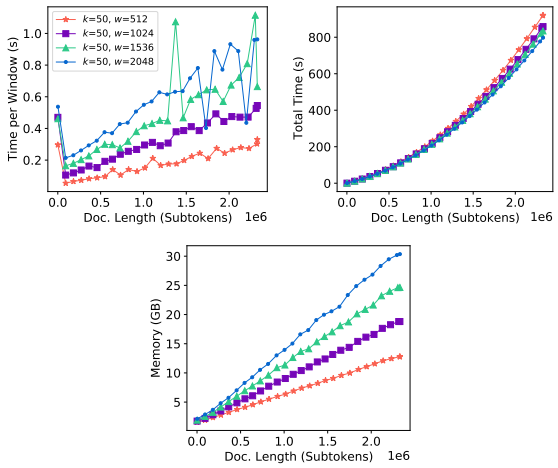

Figure 2: **Scalability to 2 million tokens** Time and memory usage of our K-NN based methods on Artaméne a book with millions of tokens.

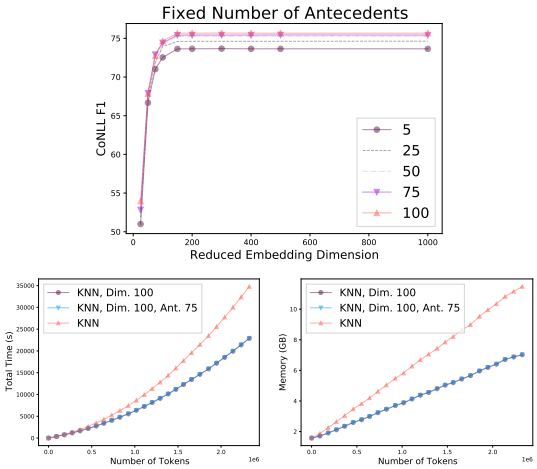

Figure 3: **Reduced Dimensionality & Candidate Antecedents**. We report CoNLL F1 on CoNLL and Time and Memory Usage on *Artamène or Cyrus the Great*.

5 Model Analysis

K-NN Sparsification Performance Analysis

In Table 3, we show the CoNLL F1 as a function of the number of neighbors k and window size w in Ours (Sp.). We observe that we can achieve high quality results even with a small number of neighbors, providing an empirical justification for our approximation. In our case, using just 10 nearest neighbors ($k = 10$) puts Ours (Sp.) within 99% of the performance of the version of our approach without sparsification. Litbank however required to use a higher value of K due to the presence of long distance coreference links in literary texts.

Dimensionality Reduction & Limiting Number of Antecedents

Window Size	512	1024	1536
k=10	77.16	77.33	77.25
k=50	77.59	77.55	77.56
k=80	77.68	77.62	77.61
k=∞	78.03	78.03	78.03

Table 3: **Sensitivity to Approximation in K-NN sparsified approach on CoNLL (F1)**.

Type	E2E (hoi)	E2E (bi)	Ours	Ours (Sp.)
Pronoun	33.74	34.57	33.62	33.50
Noun	40.59	39.98	40.36	40.00

Table 4: **Performance on mention types (F1)**.

Figure 3, shows the performance of the reduced dimensionality method 3.4 and limiting antecedents 3.5. A reduced dimension of 200 can match the performance of encoder embeddings dimension while significantly improving running time. Further, the model performs well while just keeping 75 mentions from each window of size 512(comprises squared number of possible mention spans) in the memory. We note that a reduction in memory can be achieved by dropping antecedents and using sparse matrices. However, this is not as efficient as using dense matrices on the gpu.

6 Performance Analysis

Comparisons with Baselines

To give a sense of how the proposed approximations work, we performed a simple analysis among E2E (hoi), Ours and Ours (Sp.) on the last fold of Litbank dataset. In the first experiment, we keep only the pronomial mentions in the predicted clusters and evaluate coref scores. In the second experiment we keep all mentions containing atleast one noun. Table 4 shows the final numbers. The gap in performance between E2E (hoi) and Ours seems to be equal in both categories. Ours (Sp.) and Ours have similar performance gap in both categories as well. Thus our models seem to be approximating fairly across different categories.

Table 5 shows the analysis of distance between antecedents predicted by each model on the last litbank fold. Our models seem to have a higher average distance between antecedents. This shows that the models proposed are capable to identifying long distance links. Note that distance between antecedent does not determine accuracy. A mention linked to any mention in its golden coreference cluster will have the same effect.

Type	E2E (hoi)	E2E (bi)	Ours	Ours (Sp.)
Mean	149	189.7	301.19	447.8
Max	2117	2104	2303	2303
Std.	296.8	342.9	403.01	588

Table 5: **Analysis of distance between mention and antecedent for all models in subtoken units**.

		P	R	F
CNL	Ours	78.19	77.86	78.03
	Ours - SS	78.14	77.54	77.84
	Ours - BM	78.84	77.26	78.04
Litbank	Ours	76.09	75.89	75.93
	Ours - SS	75.21	76.22	75.58
	Ours - BM	70.81	74.40	72.18
	Ours - SS - MT	68.35	68.22	68.24
MM	Ours	63.77	59.92	61.76
	Ours - SS	63.4	59.2	61.13
	Ours - BM	60.84	62.35	61.58

Table 6: **Effect of the removal of different components in Ours**. CNL - CoNLL, MM - Medmentions

Effects of model components

We further analyse the effect of other heuristics that went into the model. We use a Subtoken Strategy (SS) where we restrict the candidate mentions to align with subtoken starts, ends. As shown in Table 6, (SS) seems to have improved the results on all the datasets. Also, for litbank, mention pretraining & mention training (MT) seem to have helped significantly. Mention training forces the score of golden mentions to be higher thereby making it easy to use a threshold for singletons at inference. Bilinear mention (BM) term described in Eq.10 seemed to have helped in litbank and medmentions.

7 Related Work

With the growing computational cost of deep learning, NLP researchers have started to focus on more efficient models (Strubell et al., 2019; Schwartz et al., 2020). As coreference is a document-level phenomena, it is particularly challenging to scale, especially for long documents. While most of the work in coreference has focused on genres of text with short documents such as news articles and blogs (Pradhan et al., 2012), there has been renewed focus in long text documents such as novels (Bamman et al., 2019; Toshniwal et al., 2020b). Coreference in long text is particularly interesting due to the introduction of long-range anaphora.

Span based end-to-end coreference systems (Lee et al., 2017, 2018; Joshi et al., 2020; Wu et al., 2020) have been the state-of-the-art in short-document coreference resolution (Lu and Ng, 2020). These systems avoid training a separate mention detector. However, end-to-end coreference models are challenging to scale to long text documents due to their large memory footprint as well as slow training and inference. Thus, research on long-document coreference so far has focused on incremental (memory-based) coreference resolution (Xia et al., 2020; Toshniwal et al., 2020a,b). Memory-based approaches model coreference as online clustering by picking the most similar entity to every new mention where the cluster representations (i.e., entity representations) are also updated. However, the underlying recurrent nature of these models and the frequent read-write memory operations make these models slow. In this work, we focus on the end-to-end coreference system and show gains both in speed as well as memory.

We note that this paper's novel token-level model ideas presented in this paper were concurrently introduced by Kirstain et al. (2021). We also introduce token windows, k-nearest neighbor based sparsification techniques. Furthermore, we provide empirical results on documents with about two million tokens, which we believe to be one of the longest documents to which neural coreference models have been applied. We also note that Wu et al. (2020) hold state-of-the-art results on CoNLL (83F1). Wu et al. (2020) uses a question-answering cross-encoder style model to perform coreference. However, the method is very computationally expensive and so it is difficult to scale to the long documents which is the focus of this paper.

8 Conclusion

In this paper, we introduce a new scalable approach for performing coreference that scales to long documents. Our approach replaces costly span-based operations with token-level decisions for proposing mentions and determining antecedents. Our approach uses token similarity in the form of k-nearest neighbor graphs along with processing documents in span windows to reduce the time and memory complexity. We evaluate our proposed approach empirically and demonstrate that it achieves competitive coreference F1 scores while improving time and memory usage requirements. We demonstrate the scalability of our method by applying it to novels with about two million tokens. We further propose and demonstrate the use of low rank approximations and dropping of non essential tokens to improve memory/time efficiency.

Broader Impact and Discussion of Ethics

While our model is not tuned for any specific real-world application, our method could be used in sensitive contexts such as legal or health-care settings, and it is essential that any work using our method undertake extensive quality-assurance and robustness testing before using it in their setting. The datasets used in our work do not contain any sensitive information to the best of our knowledge.

Replicability: As part of our contributions, we will release the code used for training and evaluation in this work, as well as all the trained models at https://github.com/raghavlite/Scalable-Coreference.

References

David Bamman, Olivia Lewke, and Anya Mansoor. 2019. An annotated dataset of coreference in english literature. *arXiv preprint arXiv:1912.01140*.

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 S 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 5807–5812.

Yuval Kirstain, Ori Ram, and Omer Levy. 2021. Coreference resolution without span representations. *arXiv preprint arXiv:2101.00434*.

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.

Jing Lu and Vincent Ng. 2020. Conundrums in entity coreference resolution: Making sense of the state of the art. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6620–6631, Online. Association for Computational Linguistics.

Sunil Mohan and Donghui Li. 2019. Medmentions: a large biomedical corpus annotated with umls concepts. *arXiv preprint arXiv:1902.09476*.

Cameron Musco and Christopher Musco. 2017. Recursive sampling for the Nyström method. In *2017*, pages 3833–3845.

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.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2020. Green ai. *Commun. ACM*, 63(12):54–63.

Madeleine de Scudery. 1601. Artamenes. https://quod.lib.umich.edu/e/eebo/A70988.0001.001.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*.

Shubham Toshniwal, Allyson Ettinger, Kevin Gimpel, and Karen Livescu. 2020a. Petra: A sparsely supervised memory model for people tracking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 5415–5428. Association for Computational Linguistics.

Shubham Toshniwal, Sam Wiseman, Allyson Ettinger, Karen Livescu, and Kevin Gimpel. 2020b. Learning to ignore: Long document coreference with bounded memory neural networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8519–8526.

Christopher Williams and Matthias Seeger. 2001. Using the Nyström method to speed up kernel machines. In *2001*.

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.

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.