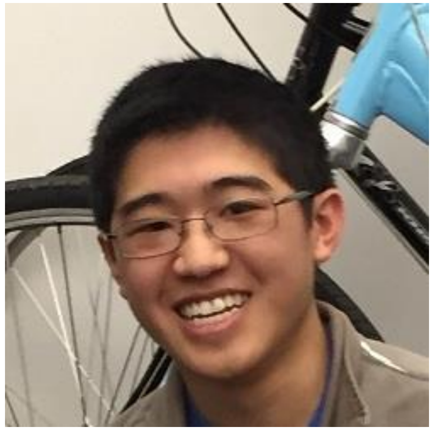
# Incremental Neural Coreference Resolution in Constant Memory

Patrick Xia, João Sedoc, Benjamin Van Durme



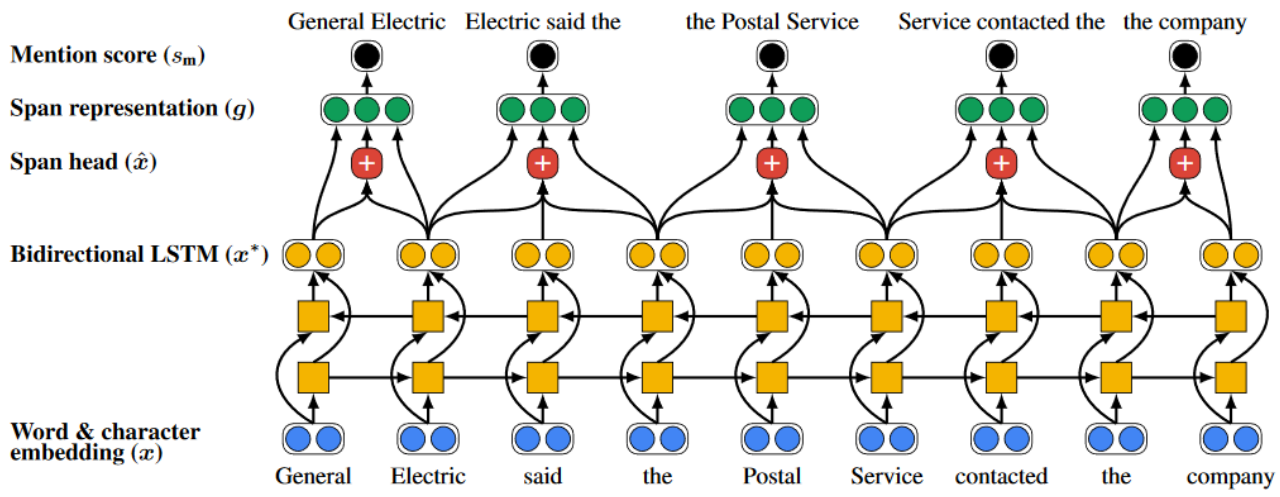
Johns Hopkins University
NYU
Center for Language and Speech Processing

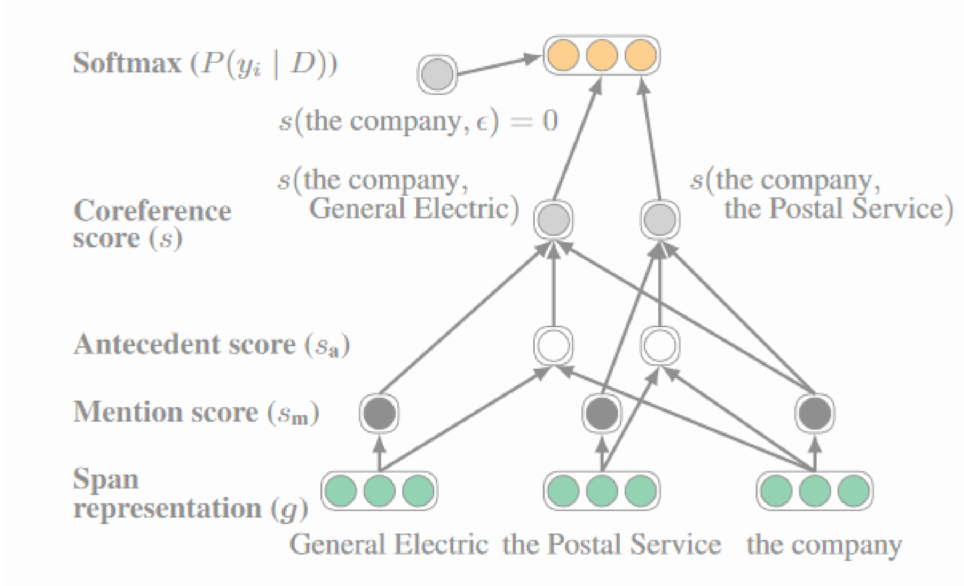
# Outline

1. Background and Motivation
2. Algorithm and Model
3. Experiments and Results

# Background

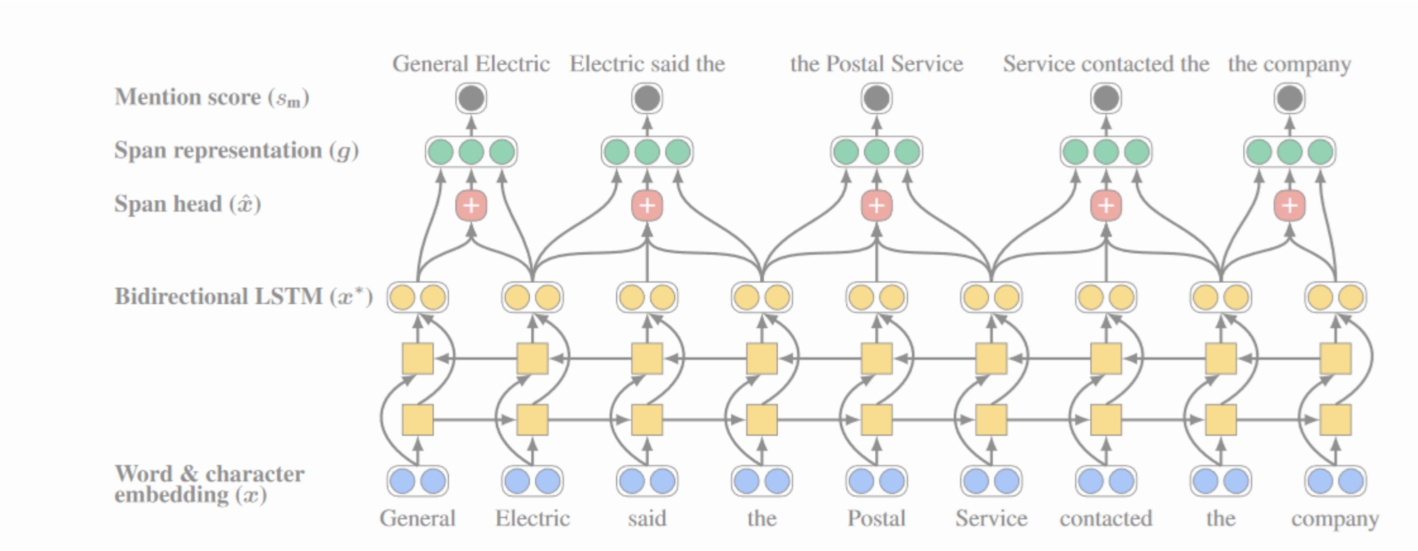
- Span Detection → Mention Pair Scoring (Lee et al., 2017)

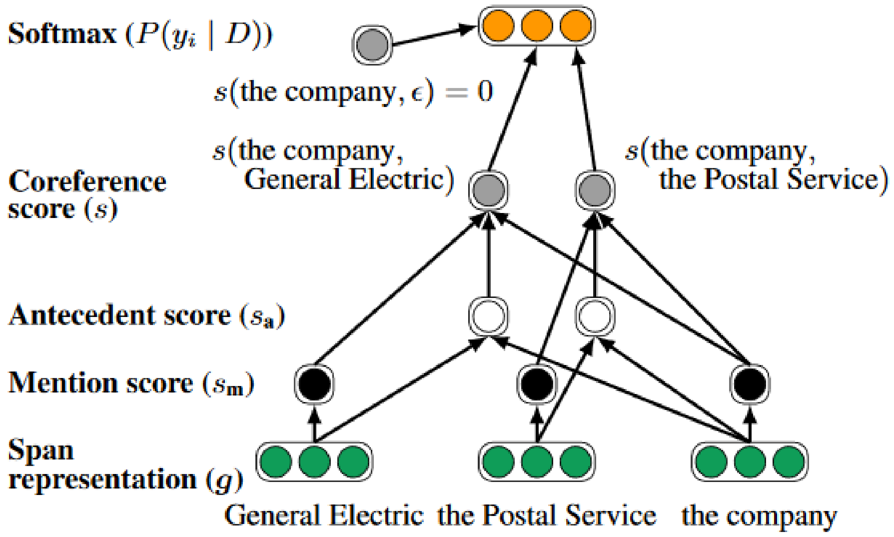


# Background

- Span Detection → Mention Pair Scoring (Lee et al., 2017)



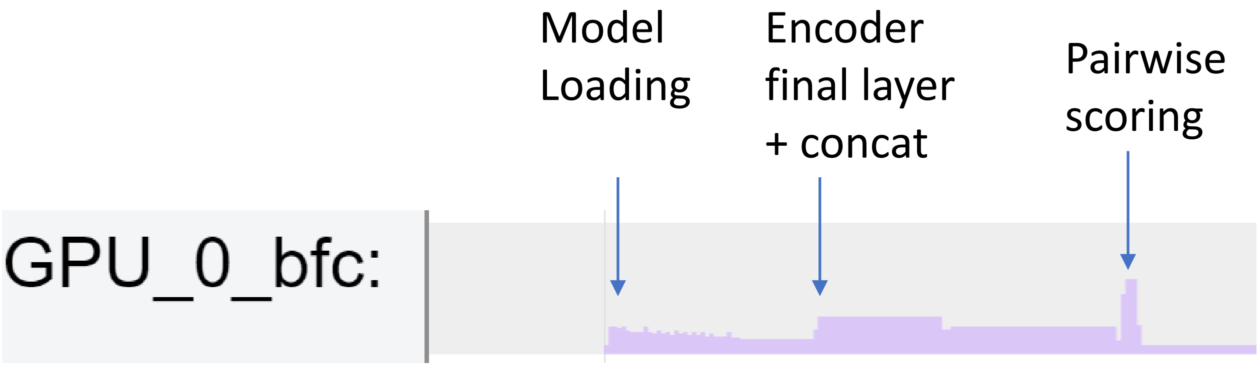
# Extensions

- Higher-order resolution: re-score against cluster average (Lee et al., 2018)
- ELMo/BERT/SpanBERT: Fine-tune pretrained encoders (Joshi et al., 2019)
- “Machine reading comprehension” scorer (Wu et al., 2020)

# Motivation

- For long documents (roughly >3000 tokens), GPUs run out of memory
  - The encoder is not always the bottleneck

- Some documents (books) exceed 100K tokens




Memory Profile for a long document with model by Joshi et al., 2019

# Some solutions…

- Some fixes:
  - Sparse Transformer
  - Sequential pairwise scoring
- … but they do not resolve:
  - Span ranking is linear in document size
  - All spans are needed in decoding
  - Even sparse Transformers need $\Omega$(document size)

# Outline

1. Background and Motivation
2. Algorithm and Model
3. Experiments and Results

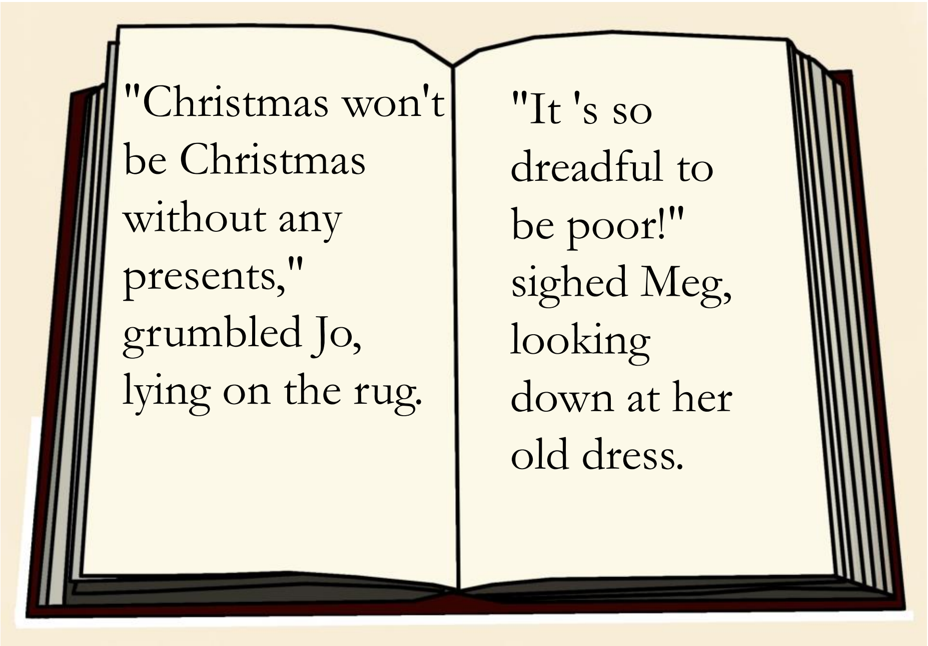
# Approach

- Limited-memory incremental algorithm for coreference resolution (Webster and Curran, 2014)
  - Similar to shift-reduce algorithms

- Neural components + **explicit** entity representations

Entity List

Document

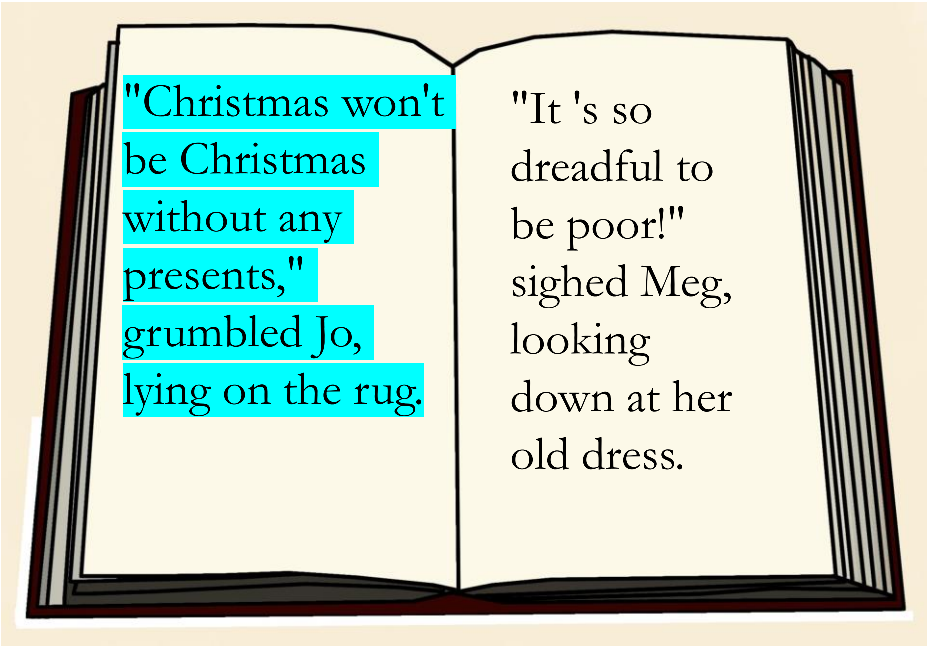



**Algorithm 1** FindClusters(Document)

Create an empty Entity List, $E$
**for** segment $\in$ Document **do**
  $M \leftarrow$ SPANS(segment)
  **for** $m \in M$ **do**
    $scores \leftarrow$ PAIRSCORE$(m, E)$
    $top\_score \leftarrow \max(scores)$
    $top\_e \leftarrow \mathrm{argmax}(scores)$
    **if** $top\_score > 0$ **then**
      UPDATE$(top_e, m)$
    **else**
      ADD_NEW_ENTITY$(E, m)$
  EVICT$(E)$
**return** $E$

Entity List

Document

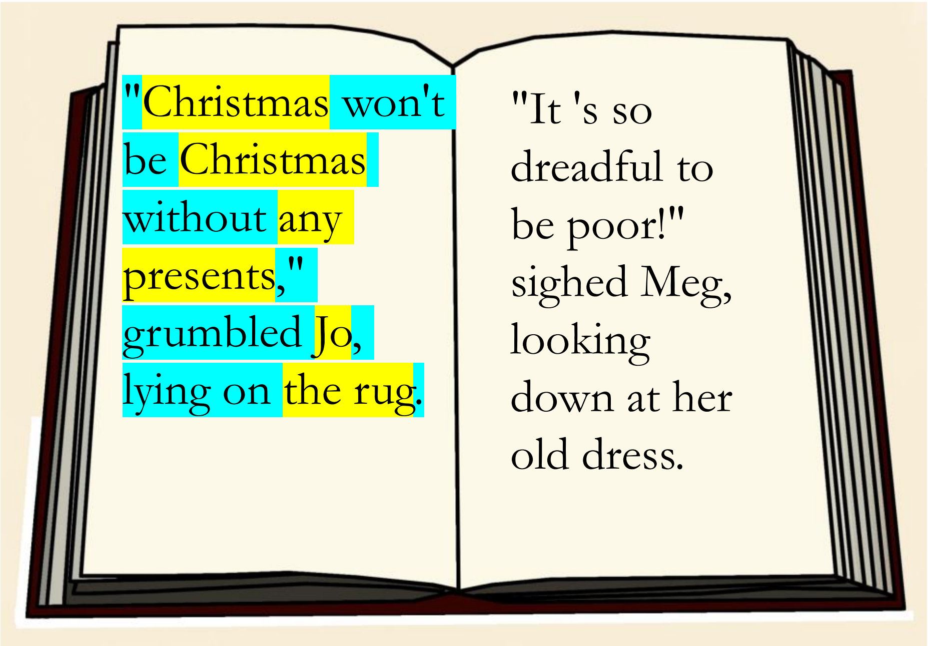



**Algorithm 1** FindClusters(Document)

Create an empty Entity List, $E$
**for** segment $\in$ Document **do**
  $M \leftarrow \textsc{Spans}(\text{segment})$
  **for** $m \in M$ **do**
    $scores \leftarrow \textsc{PairScore}(m, E)$
    $top\_score \leftarrow \max(scores)$
    $top\_e \leftarrow \operatorname{argmax}(scores)$
    **if** $top\_score > 0$ **then**
      $\textsc{Update}(top\_e, m)$
    **else**
      $\textsc{Add\_New\_Entity}(E, m)$
  $\textsc{Evict}(E)$
**return** $E$

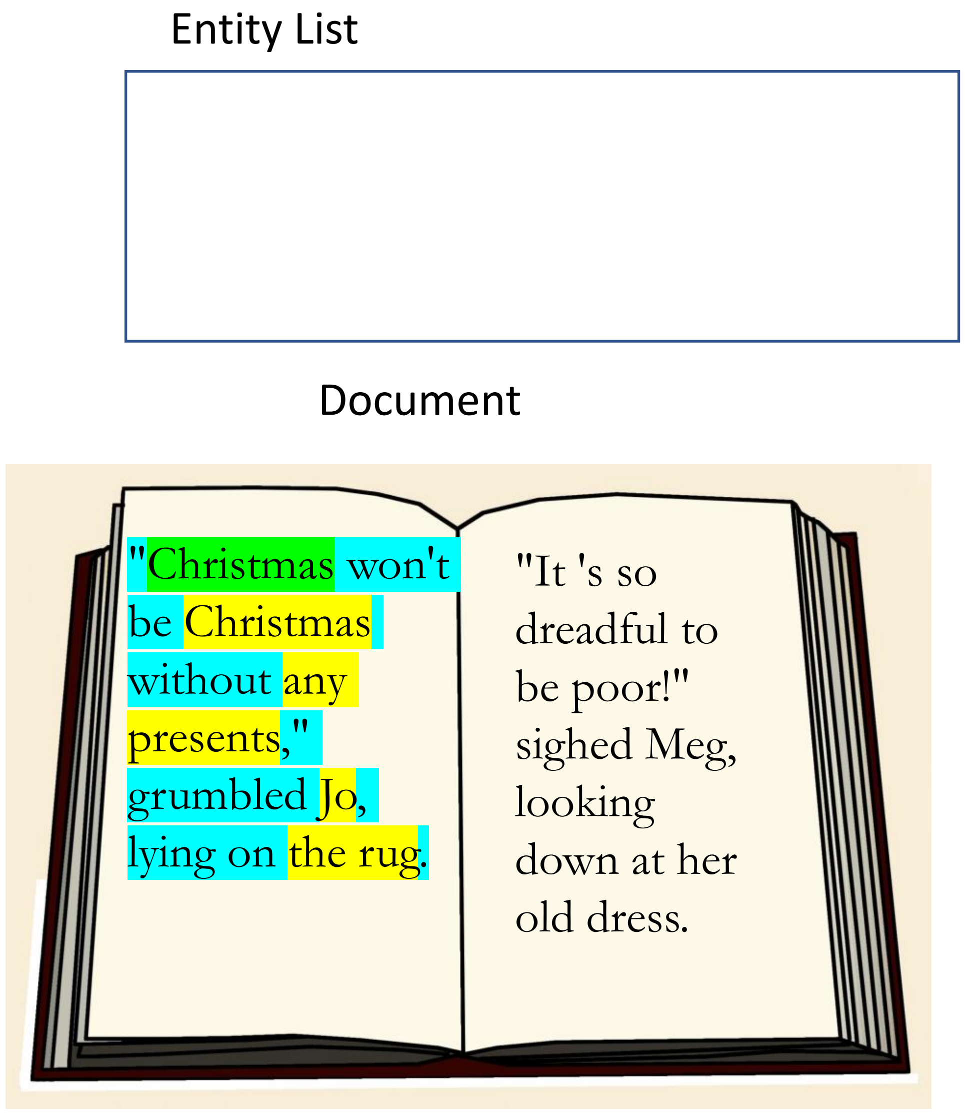
Entity List

Document




**Algorithm 1** FindClusters(Document)

Create an empty Entity List, $E$
**for** segment $\in$ Document **do**
  $M \leftarrow \textsc{Spans}(\text{segment})$
  **for** $m \in M$ **do**
    $scores \leftarrow \textsc{PairScore}(m, E)$
    $top\_score \leftarrow \max(scores)$
    $top\_e \leftarrow \mathrm{argmax}(scores)$
    **if** $top\_score > 0$ **then**
      $\textsc{Update}(top\_e, m)$
    **else**
      $\textsc{Add\_New\_Entity}(E, m)$
  $\textsc{Evict}(E)$
**return** $E$



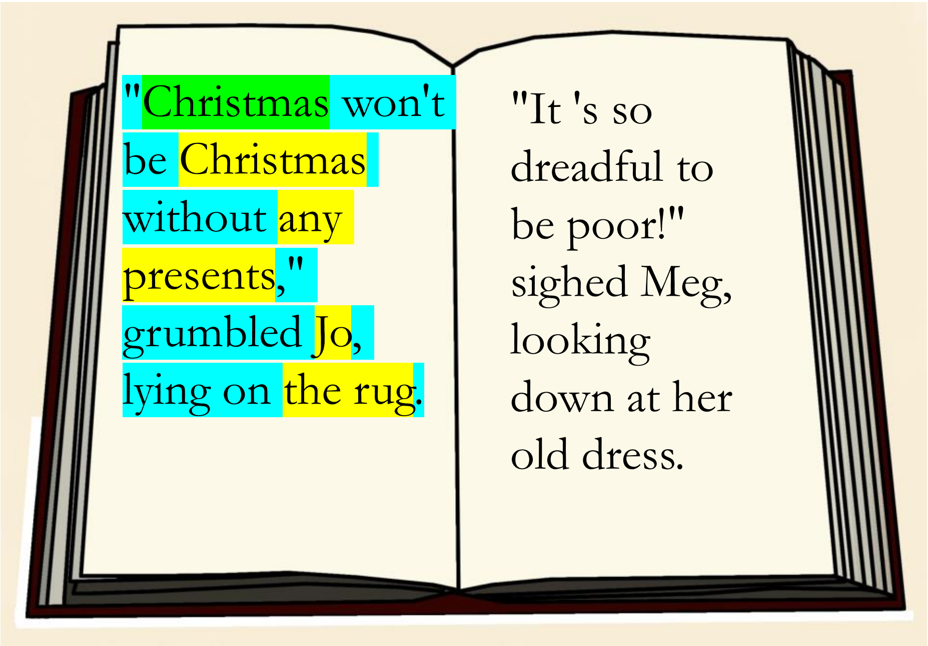

**Algorithm 1** FindClusters(Document)

Create an empty Entity List, $E$
**for** segment $\in$ Document **do**
    $M \leftarrow \textsc{Spans}(segment)$
    **for** $m \in M$ **do**
        $scores \leftarrow \textsc{PairScore}(m, E)$
        $top\_score \leftarrow \max(scores)$
        $top\_e \leftarrow \operatorname{argmax}(scores)$
        **if** $top\_score > 0$ **then**
            $\textsc{Update}(top\_e, m)$
        **else**
            $\textsc{Add\_New\_Entity}(E, m)$
    $\textsc{Evict}(E)$
**return** $E$

Entity List

{Christmas}

Document



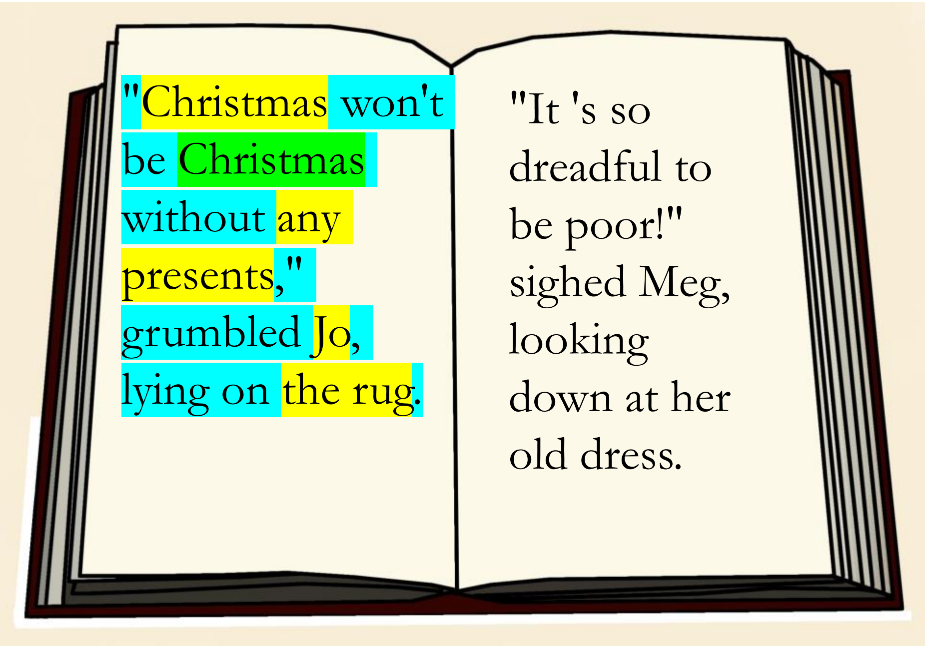

**Algorithm 1** FindClusters(Document)

Create an empty Entity List, $E$
**for** segment $\in$ Document **do**
  $M \leftarrow$ SPANS(segment)
  **for** $m \in M$ **do**
    $scores \leftarrow$ PAIRSCORE$(m, E)$
    $top\_score \leftarrow \max(scores)$
    $top\_e \leftarrow \text{argmax}(scores)$
    **if** $top\_score > 0$ **then**
      UPDATE$(top_e, m)$
    **else**
      ADD_NEW_ENTITY$(E, m)$
  EVICT$(E)$
**return** $E$

Entity List

{Christmas}

Document



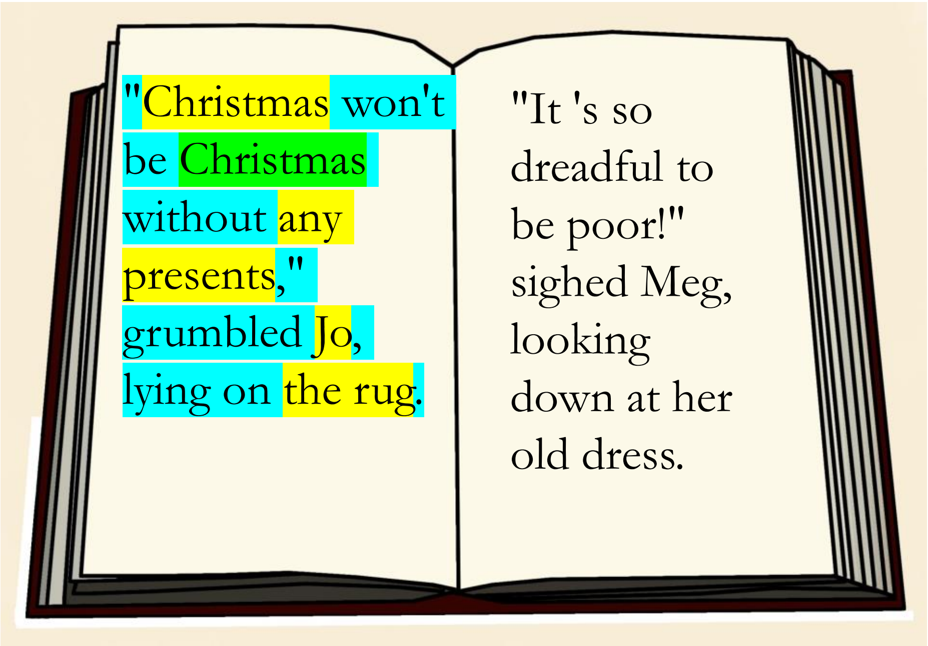

**Algorithm 1** FindClusters(Document)

Create an empty Entity List, $E$
**for** segment $\in$ Document **do**
  $M \leftarrow$ SPANS(segment)
  **for** $m \in M$ **do**
    $scores \leftarrow$ PAIRSCORE$(m, E)$
    $top\_score \leftarrow \max(scores)$
    $top\_e \leftarrow \operatorname{argmax}(scores)$
    **if** $top\_score > 0$ **then**
      UPDATE$(top_e, m)$
    **else**
      ADD_NEW_ENTITY$(E, m)$
  EVICT$(E)$
**return** $E$

Entity List

{Christmas, Christmas}

Document



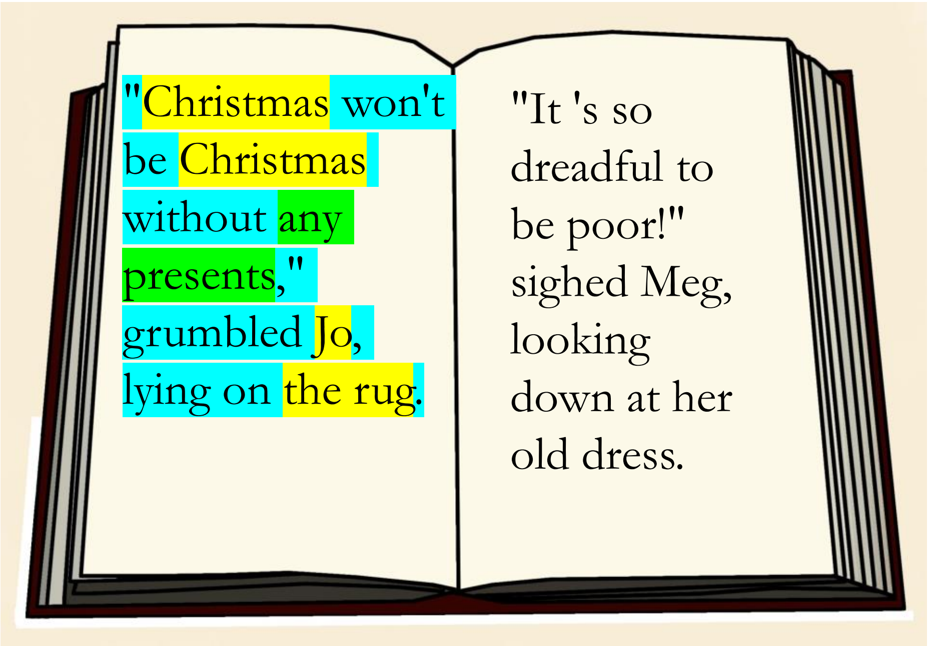

**Algorithm 1** FindClusters(Document)

Create an empty Entity List, $E$
**for** segment $\in$ Document **do**
  $M \leftarrow$ SPANS(segment)
  **for** $m \in M$ **do**
    $scores \leftarrow$ PAIRSCORE$(m, E)$
    $top\_score \leftarrow \max(scores)$
    $top\_e \leftarrow \text{argmax}(scores)$
    **if** $top\_score > 0$ **then**
      UPDATE$(top_e, m)$
    **else**
      ADD_NEW_ENTITY$(E, m)$
  EVICT$(E)$
**return** $E$

Entity List

{Christmas, Christmas}

Document



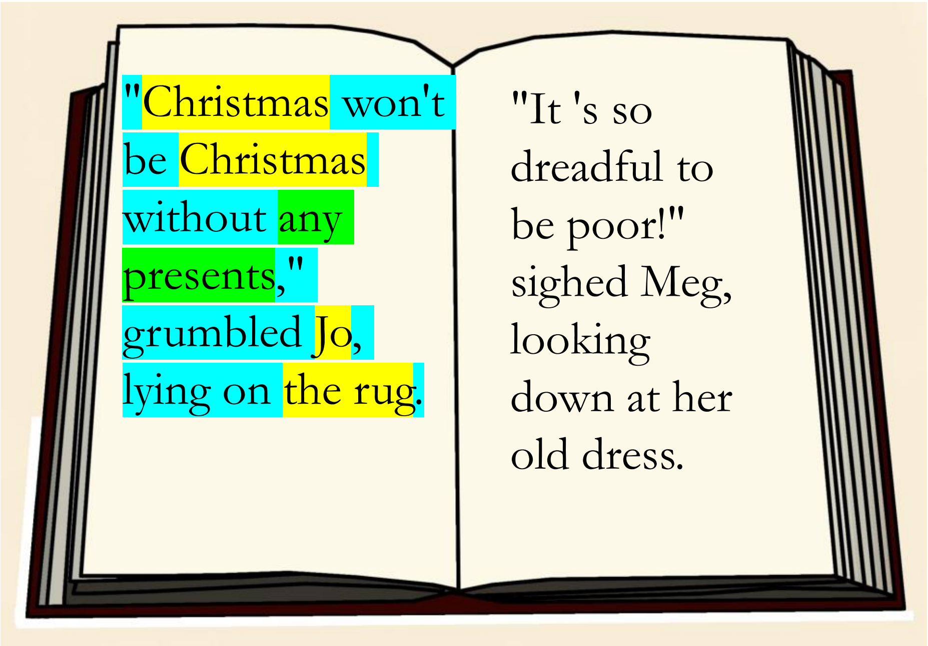

**Algorithm 1** FindClusters(Document)

Create an empty Entity List, $E$
**for** segment $\in$ Document **do**
  $M \leftarrow$ SPANS(segment)
  **for** $m \in M$ **do**
    $scores \leftarrow$ PAIRSCORE$(m, E)$
    $top\_score \leftarrow \max(scores)$
    $top\_e \leftarrow \text{argmax}(scores)$
    **if** $top\_score > 0$ **then**
      UPDATE$(top_e, m)$
    **else**
      ADD_NEW_ENTITY$(E, m)$
  EVICT$(E)$
**return** $E$

Entity List

{Christmas, Christmas},
{any presents}

Document



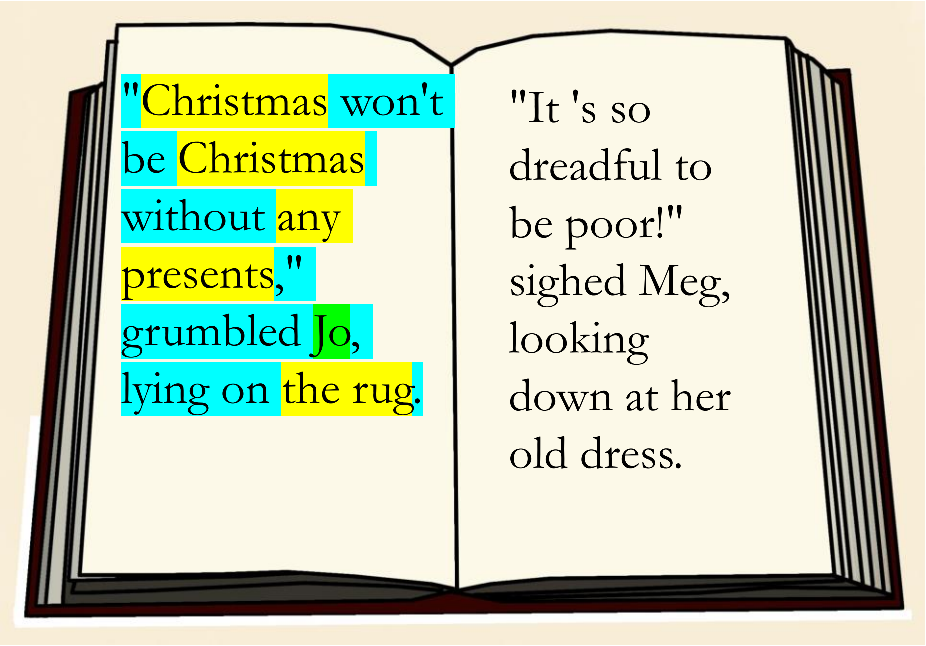

**Algorithm 1** FindClusters(Document)

Create an empty Entity List, $E$
**for** segment $\in$ Document **do**
  $M \leftarrow$ SPANS(segment)
  **for** $m \in M$ **do**
    $scores \leftarrow$ PAIRSCORE$(m, E)$
    $top\_score \leftarrow \max(scores)$
    $top\_e \leftarrow \text{argmax}(scores)$
    **if** $top\_score > 0$ **then**
      UPDATE$(top_e, m)$
    **else**
      ADD_NEW_ENTITY$(E, m)$
  EVICT$(E)$
**return** $E$

Entity List

{Christmas, Christmas},
{any presents}

Document



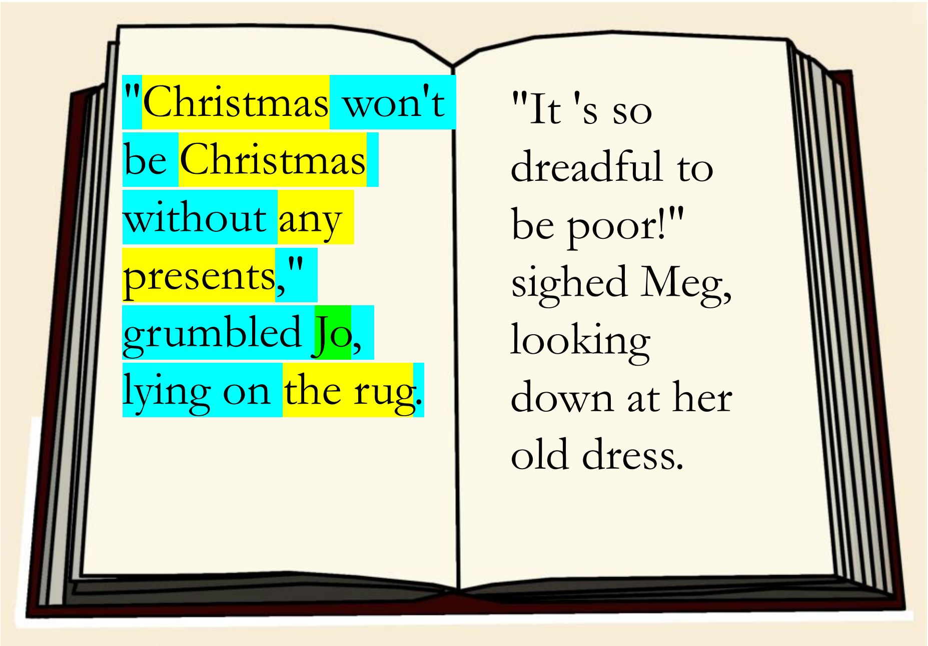

**Algorithm 1** FindClusters(Document)

Create an empty Entity List, $E$
**for** segment $\in$ Document **do**
  $M \leftarrow$ SPANS(segment)
  **for** $m \in M$ **do**
    $scores \leftarrow$ PAIRSCORE$(m, E)$
    $top\_score \leftarrow \max(scores)$
    $top\_e \leftarrow \mathrm{argmax}(scores)$
    **if** $top\_score > 0$ **then**
      UPDATE$(top_e, m)$
    **else**
      ADD_NEW_ENTITY$(E, m)$
  EVICT$(E)$
**return** $E$

Entity List

{Christmas, Christmas},
{any presents},
{Jo}

Document



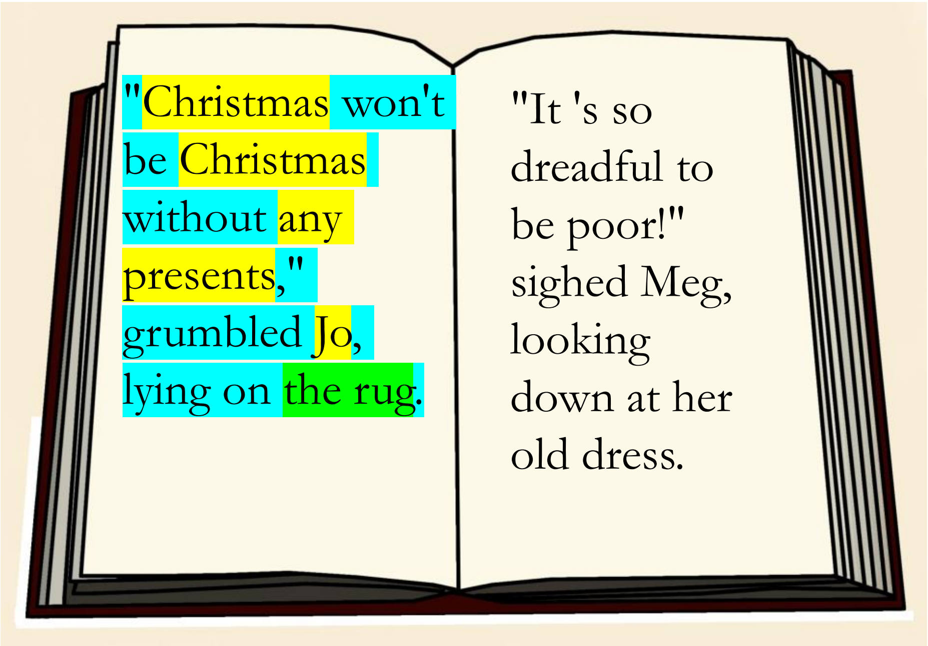

**Algorithm 1** FindClusters(Document)

Create an empty Entity List, $E$
**for** segment $\in$ Document **do**
  $M \leftarrow$ SPANS(segment)
  **for** $m \in M$ **do**
    $scores \leftarrow$ PAIRSCORE$(m, E)$
    $top\_score \leftarrow \max(scores)$
    $top\_e \leftarrow \operatorname{argmax}(scores)$
    **if** $top\_score > 0$ **then**
      UPDATE$(top_e, m)$
    **else**
      ADD_NEW_ENTITY$(E, m)$
  EVICT$(E)$
**return** $E$

Entity List

{Christmas, Christmas},
{any presents},
{Jo}

Document



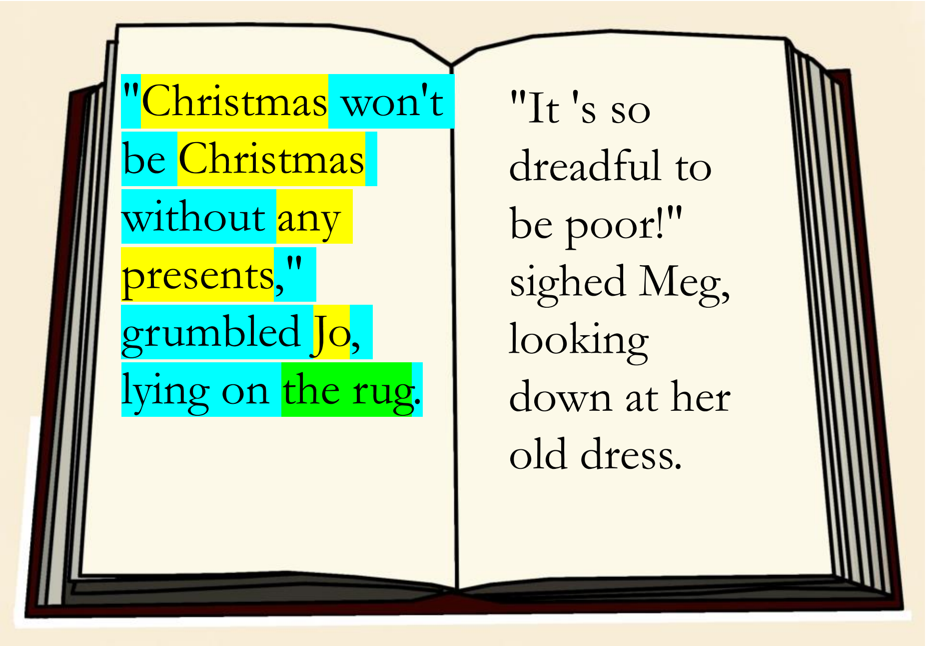

**Algorithm 1** FindClusters(Document)

Create an empty Entity List, $E$
**for** segment $\in$ Document **do**
  $M \leftarrow$ SPANS(segment)
  **for** $m \in M$ **do**
    $scores \leftarrow$ PAIRSCORE$(m, E)$
    $top\_score \leftarrow \max(scores)$
    $top\_e \leftarrow \text{argmax}(scores)$
    **if** $top\_score > 0$ **then**
      UPDATE$(top_e, m)$
    **else**
      ADD_NEW_ENTITY$(E, m)$
  EVICT$(E)$
**return** $E$

Entity List

{Christmas, Christmas},
{any presents},
{Jo},
{the rug}

Document



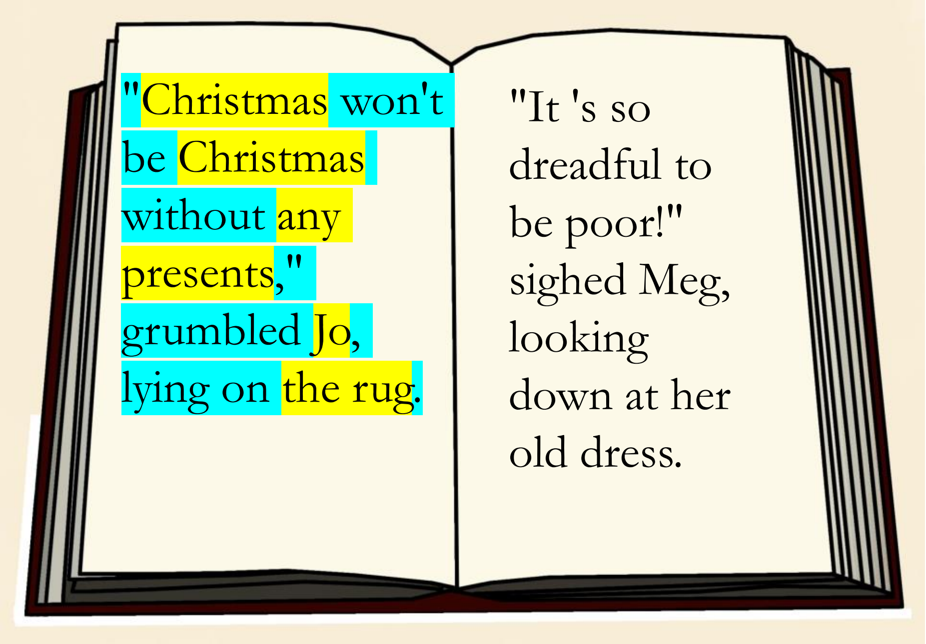

**Algorithm 1** FindClusters(Document)

Create an empty Entity List, $E$
**for** segment $\in$ Document **do**
  $M \leftarrow \textsc{Spans}(\text{segment})$
  **for** $m \in M$ **do**
    $scores \leftarrow \textsc{PairScore}(m, E)$
    $top\_score \leftarrow \max(scores)$
    $top\_e \leftarrow \operatorname{argmax}(scores)$
    **if** $top\_score > 0$ **then**
      $\textsc{Update}(top\_e, m)$
    **else**
      $\textsc{Add\_New\_Entity}(E, m)$
  $\textsc{Evict}(E)$
**return** $E$

Entity List

{Christmas, Christmas},
{any presents},
{Jo},
{the rug}

Document



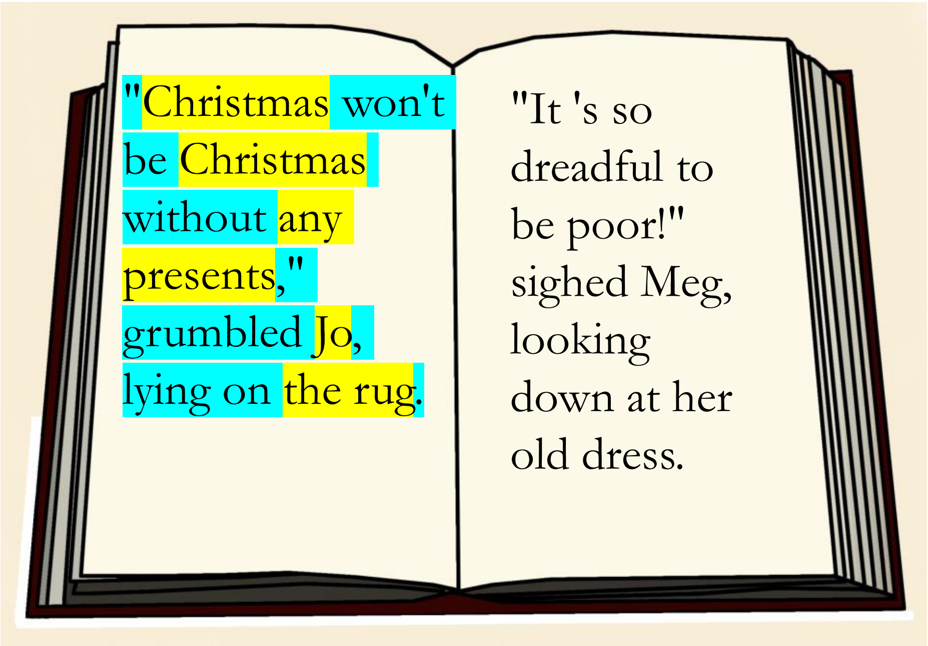

**Algorithm 1** FindClusters(Document)

Create an empty Entity List, $E$
**for** segment $\in$ Document **do**
  $M \leftarrow$ SPANS(segment)
  **for** $m \in M$ **do**
    $scores \leftarrow$ PAIRSCORE$(m, E)$
    $top\_score \leftarrow \max(scores)$
    $top\_e \leftarrow \mathrm{argmax}(scores)$
    **if** $top\_score > 0$ **then**
      UPDATE$(top_e, m)$
    **else**
      ADD_NEW_ENTITY$(E, m)$
  EVICT$(E)$
**return** $E$

Entity List

{Christmas, Christmas},
~~{any presents},~~
{Jo},
~~{the rug}~~

Document



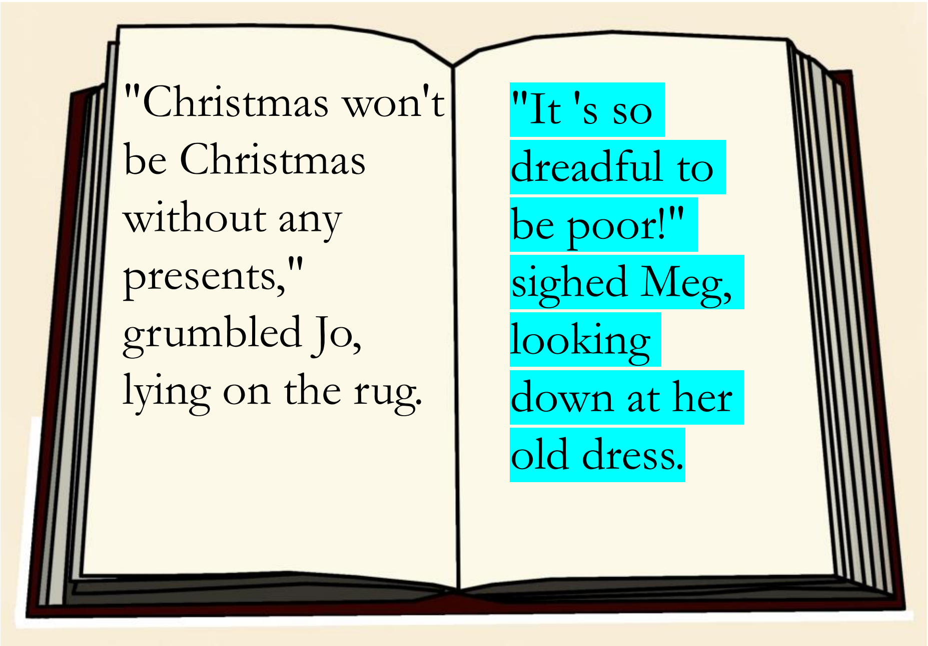

**Algorithm 1** FindClusters(Document)

Create an empty Entity List, $E$
**for** segment $\in$ Document **do**
  $M \leftarrow \textsc{Spans}(\text{segment})$
  **for** $m \in M$ **do**
    $scores \leftarrow \textsc{PairScore}(m, E)$
    $top\_score \leftarrow \max(scores)$
    $top\_e \leftarrow \text{argmax}(scores)$
    **if** $top\_score > 0$ **then**
      $\textsc{Update}(top\_e, m)$
    **else**
      $\textsc{Add\_New\_Entity}(E, m)$
  $\textsc{Evict}(E)$
**return** $E$

Entity List

{Christmas, Christmas},
{Jo},

Document




**Algorithm 1** FindClusters(Document)

Create an empty Entity List, $E$
**for** segment $\in$ Document **do**
    $M \leftarrow$ SPANS(segment)
    **for** $m \in M$ **do**
        $scores \leftarrow$ PAIRSCORE$(m, E)$
        $top\_score \leftarrow \max(scores)$
        $top\_e \leftarrow \mathrm{argmax}(scores)$
        **if** $top\_score > 0$ **then**
            UPDATE$(top_e, m)$
        **else**
            ADD_NEW_ENTITY$(E, m)$
    EVICT$(E)$
**return** $E$

# Implementation

- Encoder: SpanBERT
- **Spans**: top-*k* spans based on learned scorer
- **PairScore**: FFNN(m, e)
- **Update**: learned average of spans
- **Evict**: all old entities

**Algorithm 1** FindClusters(Document)

Create an empty Entity List, $E$
**for** segment $\in$ Document **do**
    $M \leftarrow$ SPANS(segment)
    **for** $m \in M$ **do**
        $scores \leftarrow$ PAIRSCORE$(m, E)$
        $top\_score \leftarrow \max(scores)$
        $top\_e \leftarrow \text{argmax}(scores)$
        **if** $top\_score > 0$ **then**
            UPDATE$(top_e, m)$
        **else**
            ADD_NEW_ENTITY$(E, m)$
    EVICT$(E)$
**return** $E$

# Outline

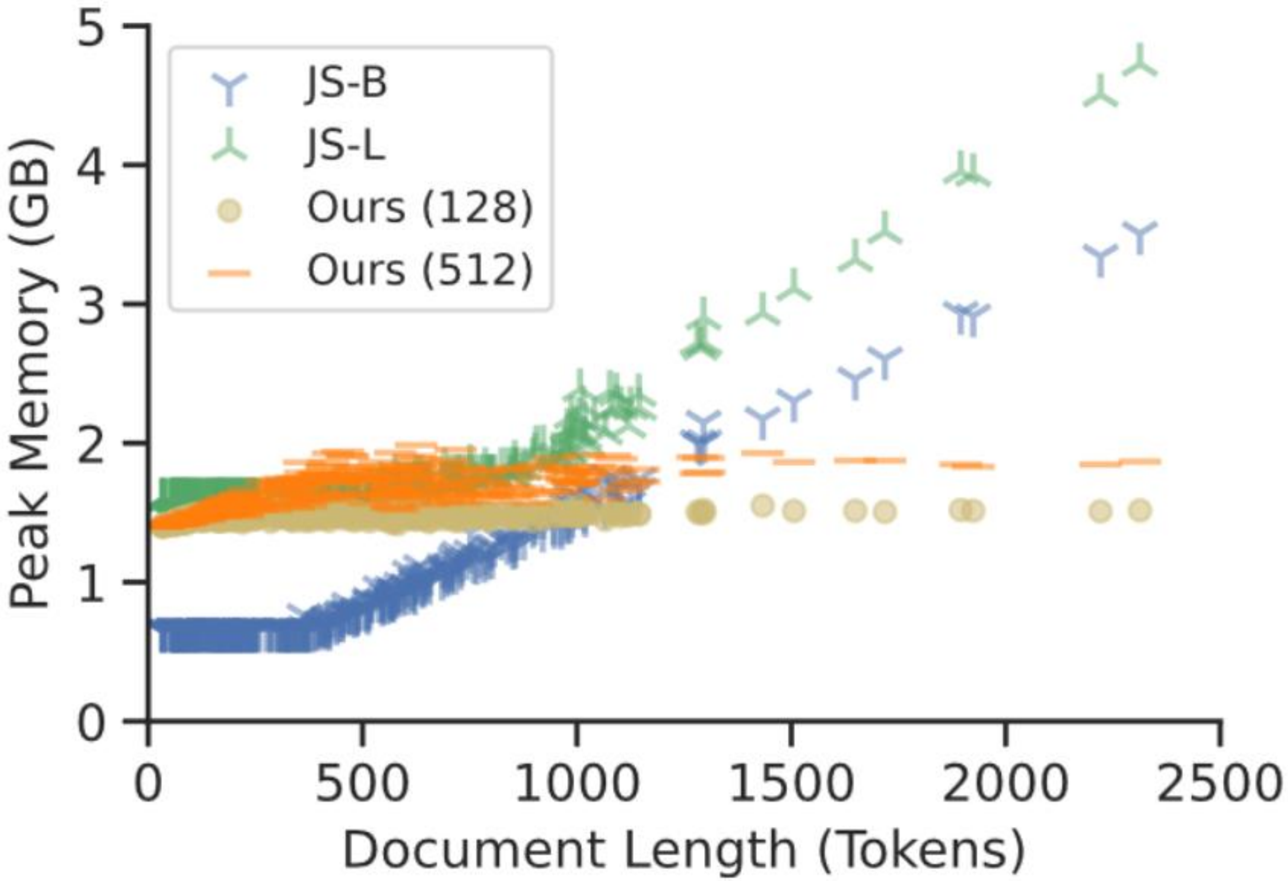
1. Background and Motivation
2. Algorithm and Model
3. Experiments and Results

# Experiments

- OntoNotes 5.0 (English)
- Evaluated with average F1 (MUC, $B^3$, and $CEAF_{\phi 4}$)
- Goal: compare performance between incremental algorithm vs. full-document model

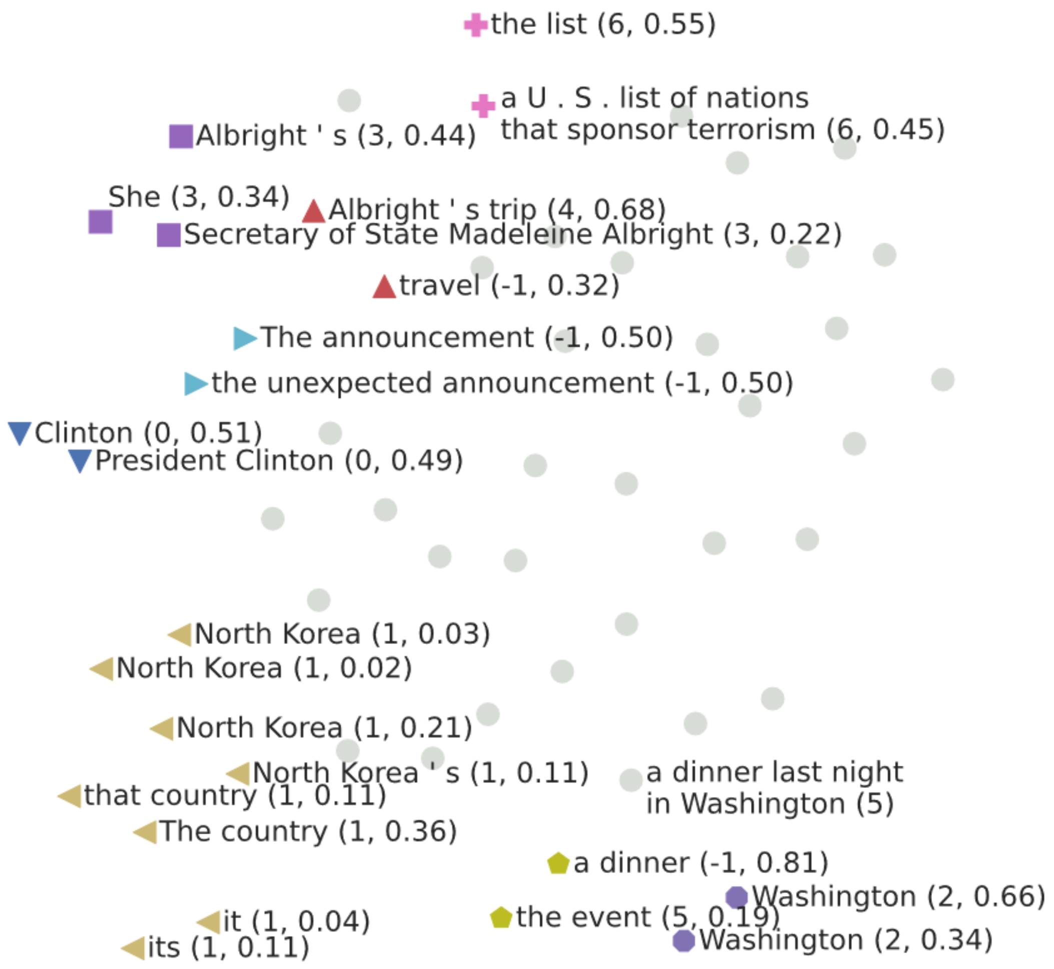
# Results

- 79.6 F1 → 79.4 F1
  - Virtually no loss in performance
- Constant space implementation at inference (2GB) and training (<10GB)
- Our algorithm has O(1) space complexity
  - A fixed-sized set of entities kept across time



# Entity Analysis

- Coreference is well-suited for (online) clustering: coreferent mentions are close in embedding space



# Conclusions

- Constant memory algorithm + model for coreference resolution
- Can be applied to future SOTA models
- Detailed analysis of document and segment lengths in paper!
- Thanks!

Code and models at
github.com/pitrack/incremental-coref