Incremental Neural Coreference Resolution in Constant Memory

Patrick Xia, João Sedoc, Benjamin Van Durme
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 $\rightarrow$ 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
"Christmas won't be Christmas without any presents," grumbled Jo, lying on the rug.

"It's so dreadful to be poor!" sighed Meg, looking down at her old dress.
"Christmas won't be Christmas without any presents," grumbled Jo, lying on the rug.

"It's so dreadful to be poor!" sighed Meg, looking down at her old dress.

**Algorithm 1 FindClusters(Document)**

Create an empty Entity List, $E$

for segment ∈ Document do

$M \leftarrow \text{SPANS}(\text{segment})$

for $m \in M$ do

scores $\leftarrow \text{PAIRSCORE}(m, E)$

top_score $\leftarrow \max(\text{scores})$

top_e $\leftarrow \text{argmax}(\text{scores})$

if top_score $> 0$ then

UPDATE(top_e, m)

else

ADD_NEW_ENTITY($E, m$)

EVICT($E$)

return $E$
"Christmas won't be Christmas without any presents," grumbled Jo, lying on the rug.

"It 's so dreadful to be poor!" sighed Meg, looking down at her old dress.
"Christmas won't be Christmas without any presents," grumbled Jo, lying on the rug. "It's so dreadful to be poor!" sighed Meg, looking down at her old dress.
"Christmas won't be Christmas without any presents," grumbled Jo, lying on the rug. "It's so dreadful to be poor!" sighed Meg, looking down at her old dress.
"Christmas won't be Christmas without any presents," grumbled Jo, lying on the rug. "It's so dreadful to be poor!" sighed Meg, looking down at her old dress.
"Christmas won't be Christmas without any presents," grumbled Jo, lying on the rug. "It's so dreadful to be poor!" sighed Meg, looking down at her old dress.
"Christmas won't be Christmas without any presents," grumbled Jo, lying on the rug. "It's so dreadful to be poor!" sighed Meg, looking down at her old dress.
"Christmas won't be Christmas without any presents," grumbled Jo, lying on the rug. "It's so dreadful to be poor!" sighed Meg, looking down at her old dress.
"Christmas won't be Christmas without any presents," grumbled Jo, lying on the rug. "It's so dreadful to be poor!" sighed Meg, looking down at her old dress.
"Christmas won't be Christmas without any presents," grumbled Jo, lying on the rug.

"It's so dreadful to be poor!" sighed Meg, looking down at her old dress.
"Christmas won't be Christmas without any presents," grumbled Jo, lying on the rug. "It's so dreadful to be poor!"

sighed Meg, looking down at her old dress.

Algorithm 1 FindClusters(Document)

Create an empty Entity List, $E$

for segment \(\in\) Document do

$M \leftarrow \text{SPANS}(\text{segment})$

for \(m \in M\) do

scores \(\leftarrow\) PAIRSORE($m, E$)

top_score \(\leftarrow\) max(scores)

top_e \(\leftarrow\) argmax(scores)

if top_score > 0 then

UPDATE(top_e, m)

else

ADD_NEW_ENTITY($E, m$)

Evict($E$)

return $E$
"Christmas won't be Christmas without any presents," grumbled Jo, lying on the rug. "It's so dreadful to be poor!" sighed Meg, looking down at her old dress.
"Christmas won't be Christmas without any presents," grumbled Jo, lying on the rug. "It's so dreadful to be poor!" sighed Meg, looking down at her old dress.
"Christmas won't be Christmas without any presents," grumbled Jo, lying on the rug.

"It's so dreadful to be poor!" sighed Meg, looking down at her old dress.

"Christmas won't be Christmas without any presents," grumbled Jo, lying on the rug.

"It's so dreadful to be poor!" sighed Meg, looking down at her old dress.
"Christmas won't be Christmas without any presents," grumbled Jo, lying on the rug. "It's so dreadful to be poor!" sighed Meg, looking down at her old dress.
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** \text{FindClusters}(Document)

Create an empty Entity List, $E$

\begin{verbatim}
for \ segment \in \ Document \ do \\
M \leftarrow \text{SPANS}(\text{segment}) \\
for \ m \in M \ do \\
\text{scores} \leftarrow \text{PAIRSCORE}(m, E) \\
\text{top}\_\text{score} \leftarrow \max(\text{scores}) \\
\text{top}\_e \leftarrow \argmax(\text{scores}) \\
if \text{top}\_\text{score} > 0 \ then \\
\quad \text{UPDATE}(\text{top}\_e, m) \\
else \\
\quad \text{ADD\_NEW\_ENTITY}(E, m) \\
\quad \text{EVICT}(E) \\
return E
\end{verbatim}
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