Character Coreference Resolution in Movie Screenplays

Sabyasachee Baruah
University of Southern California
sbaruah@usc.edu

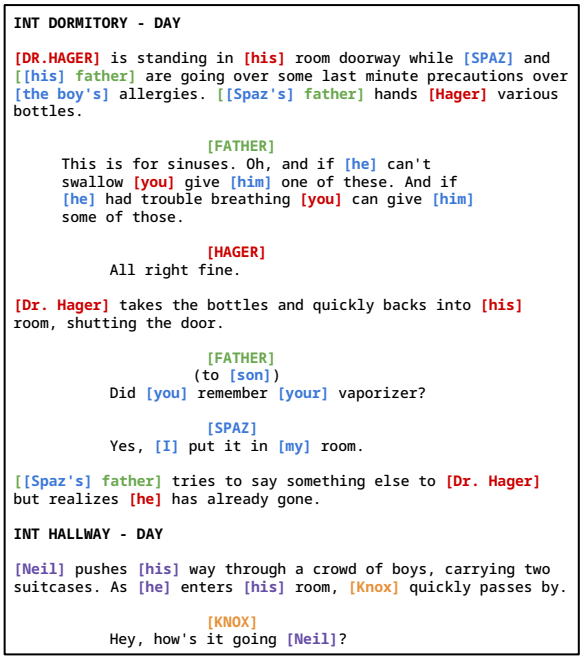
Shrikanth Narayanan
University of Southern California
shri@ee.usc.edu

Abstract

Movie screenplays have a distinct narrative structure. It segments the story into scenes containing interleaving descriptions of actions, locations, and character dialogues. A typical screenplay spans several scenes and can include long-range dependencies between characters and events. A holistic document-level understanding of the screenplay requires several natural language processing capabilities, such as parsing, character identification, coreference resolution, action recognition, summarization, and attribute discovery. In this work, we develop scalable and robust methods to extract the structural information and character coreference clusters from full-length movie screenplays. We curate two datasets for screenplay parsing and character coreference — *MovieParse* and *MovieCoref*, respectively. We build a robust screenplay parser to handle inconsistencies in screenplay formatting and leverage the parsed output to link co-referring character mentions. Our coreference models can scale to long screenplay documents without drastically increasing their memory footprints. We open-source our model and datasets here.[1]

1 Introduction

Screenplays are semi-structured text documents that narrate the events within the story of a movie, TV show, theatrical play, or other creative media. It contains stage directions that provide a blueprint for the produced content, offering valuable information for media understanding tasks (Turetsky and Dimitrova, 2004; Sap et al., 2017; Martinez et al., 2022). Automated text processing methods can enable a scalable and nuanced analysis of screenplays and provide key novel insights about the narrative elements and story characteristics (Ramakrishna et al., 2017). This paper focuses on a critical aspect of automatic screenplay analysis: character coreference resolution.


INT DORMITORY - DAY

[DR.HAGER] is standing in [his] room doorway while [SPAZ] and [[his] father] are going over some last minute precautions over [the boy's] allergies. [[Spaz's] father] hands [Hager] various bottles.

[FATHER]
This is for sinuses. Oh, and if [he] can't swallow [you] give [him] one of these. And if [he] had trouble breathing [you] can give [him] some of those.

[HAGER]
All right fine.

[Dr. Hager] takes the bottles and quickly backs into [his] room, shutting the door.

[FATHER]
(to [son])
Did [you] remember [your] vaporizer?

[SPAZ]
Yes, [I] put it in [my] room.

[[Spaz's] father] tries to say something else to [Dr. Hager] but realizes [he] has already gone.

INT HALLWAY - DAY

[Neil] pushes [his] way through a crowd of boys, carrying two suitcases. As [he] enters [his] room, [Knox] quickly passes by.

[KNOX]
Hey, how's it going [Neil]?

Figure 1: An example screenplay excerpt from the movie *Dead Poets Society (1989)*. Like-colored mentions are co-referring.

A screenplay has a hierarchical structure that includes a sequence of scenes, with each scene containing action passages and character utterances(Argentini, 1998). A scene starts with a *slugline* that specifies where and when the scene takes place. The action passages introduce the characters and describe their actions, and the utterances contain their verbal interactions. Sometimes, transition phrases, such as *FADE IN* and *CUT OUT*, separate adjacent scenes, detailing the camera movement. These structural elements form the document-level syntax of the screenplay. Identifying this modular structure, called screenplay parsing, is an essential preprocessing step for downstream analyses.

Screenwriters usually follow a uniform indentation and word case scheme while formatting the screenplay. Some standard conventions are

[1]MovieParse: https://github.com/usc-sail/movie_screenplay_parser, MovieCoref: https://github.com/usc-sail/movie_character_coref

Findings of the Association for Computational Linguistics: ACL 2023, pages 10300–10313
July 9-14, 2023

to arrange sluglines and action passages at the same indentation level and write sluglines and speaker names in upper case (Riley, 2009). However, publicly-available screenplays can exhibit wide variability in document formatting and deviate from these norms by removing indentations, containing watermark characters, or omitting location keywords (*INT* and *EXT*) in sluglines. A screenplay parser should be robust to these issues to extract the document's structure correctly and consistently. Once parsed, we can process the extracted structural segments for narrative analysis.

Narrativity occurs when characters interact with each other in some spatiotemporal context (Piper et al., 2021). Computational narrative understanding involves natural language tasks such as named entity recognition (NER) to identify the characters (Srivastava et al., 2016), coreference resolution to gather co-referring character mentions (Elson et al., 2010; Baruah et al., 2021), semantic role labeling to find their actions (Martinez et al., 2022), relation extraction to discover character attributes (Yu et al., 2022), sentiment analysis to understand the attitude of their interactions (Nalisnick and Baird, 2013), and summarization to spot the key events (Gorinski and Lapata, 2015). Out of these tasks, coreference resolution presents a unique challenge because of the long document size of screenplays (Baruah et al., 2021). Modern coreference models rely on transformers to capture the discourse semantics. However, the average screenplay length (30K words) far exceeds the restricted context of transformers (512 tokens). As document size increases, the number of words between a mention and its antecedent increases. Several scenes can elapse before a character is mentioned again in the screenplay. A screenplay coreference model should be able to handle such distant coreference relations. Fig 1 shows an example of a screenplay excerpt annotated with character coreference labels.

In this paper, we tackle character coreference resolution in movie screenplays. We focus on characters because they are the primary agents that drive the plot (Bamman et al., 2013) through rich and dynamic interactions (Labatut and Bost, 2019). Long-range coreference relations are also more common for characters than other entities (Bamman et al., 2020). To support our modeling, we augment existing screenplay datasets synthetically and through human annotations. First, we systematically add different formatting perturbations to screenplay documents and train our screenplay parser to be robust to these variations. We use this parser to find speakers and segment boundaries as a preprocessing step to coreference resolution. Second, we annotate character mentions in six full-length screenplays and model the coreference relation by scoring word pairs (Dobrovolskii, 2021). We adapt the model inference to long screenplay documents by fusion-based and hierarchical methods. We summarize our contributions as follows:

1. We curate and share two screenplay datasets called *MovieParse* and *MovieCoref* for movie screenplay parsing and character coreference resolution, respectively.

2. We develop a robust screenplay parser to extract the screenplay structure. It can handle various screenplay formatting styles.

3. We build a character coreference model and adapt it to long screenplay documents.

2 Related Work

Screenplay Parsing. Van Rijsselbergen et al. (2009) used unified modeling language to represent the structural elements of the screenplay document. Agarwal et al. (2014) trained support vector machines on synthetic training data to build a robust screenplay parser. We adopt a similar approach, but handle a wider set of document-related issues and leverage modern sequence embedding models instead of hand-crafted features. Winer and Young (2017) used a recursive descent parser to extract the spatiotemporal information from sluglines. Ramakrishna et al. (2017) built a rule-based screenplay parser to find character names and utterances, ignoring the action passages. Baruah et al. (2021) annotated the line-wise structural tag of 39 screenplay excerpts to evaluate their rule-based parser. We extend this dataset with synthetic formatting variations to train our robust screenplay parser.

Screenplay Coreference Resolution. Baruah et al. (2021) established annotation guidelines for the character coreference task in screenplays. They annotated three screenplay excerpts to evaluate pretrained coreference models. They combined the neural model with rules inspired by the narrative structure of screenplays. The limitation of their work is that they used excerpts instead of full-length scripts. We adopt their annotation guidelines to label six full-length screenplays, enabling us to study how our models scale to the entire narrative.

Literary Coreference Resolution. Several past studies have tried to extract social networks from literary texts to study character interactions, where they naturally need to unify different character mentions to create the network's nodes (Labatut and Bost, 2019). Most methods clustered the person names using heuristics such as matching gender and honorifics or finding name variations and nicknames (Elson et al., 2010; Elsner, 2012; Coll Ardanuy and Sporleder, 2014; Vala et al., 2015). Bamman et al. (2019) and Bamman et al. (2020) created the LitBank corpus containing entity, event, and coreference annotations of 100 works of English-language fiction and trained an end-to-end neural coreference model. Yoder et al. (2021) developed a text processing pipeline for fan-fiction stories where they used Joshi et al.'s (2020) SpanBERT model to find character coreference clusters.

Coreference Resolution. Lee et al.'s (2017) seminal work on end-to-end training of neural coreference models was a significant breakthrough in coreference resolution. Lee et al. (2018) later improved the model's efficiency by splitting the coreference scoring into coarse and fine stages. Joshi et al. (2019) and Joshi et al. (2020) replaced the encoder with BERT-based architectures. Reducing time and memory footprints became imperative with the introduction of transformer-based encoders. Kirstain et al. (2021) bypassed creating explicit span representations and modeled each span by the word embeddings of its boundary tokens. Combined with mention pruning, they achieved quadratic complexity in document size. Dobrovolskii (2021) substituted spans with head words, removing the need for mention pruning and maintained the same quadratic runtime. Bohnet et al. (2022) used a text-to-text paradigm to make sentence-level co-referential decisions and achieved state-of-the-art performance using the T5 text-to-text encoder (Raffel et al., 2022).

However, these methods do not scale to long documents because the quadratic complexity of scoring mention (span or token) pairs becomes intractable as document size increases.[2] Memory-bounded methods (Xia et al., 2020; Toshniwal et al., 2020; Thirukovalluru et al., 2021) keep a finite set of entity representations and update it incrementally for each mention. Most of these models are evaluated on the OntoNotes (Pradhan et al., 2012) and LitBank corpora whose average document length is less than 500 and 2K tokens, respectively: an order of magnitude less than the average screenplay size. The entity spread (number of tokens between the first and last mention) and the maximum active entity count (active entity count of a token is the number of entities whose spread includes the token) are larger for screenplays because the main characters tend to appear throughout the story in bursts (Bamman et al., 2020). In this work, we adapt Dobrovolskii's (2021) word-level coreference model to movie screenplays by fusing the word-pair coreference scores from overlapping segments or by running inference hierarchically.

3 Screenplay Parsing

3.1 Problem Setup

Agarwal et al. (2014) posed screenplay parsing as a classification task of assigning a single structural label to each screenplay line. The structural types include slugline, action, speaker, expression, utterance, and transition. We define a screenplay *segment* as a contiguous sequence of lines with the same structural label.

Sluglines indicate the beginning of a new scene. They contain information about the scene's location, time, and whether it occurs in an interior (INT) or exterior (EXT) setting. **Action** lines describe the characters and their actions. Most non-dialogue lines fall under this class. **Speaker** lines contain character names that immediately precede their utterances. **Utterance** lines comprise words spoken by the characters. **Expression** lines describe speech acts or other scene information that provide more context to utterances, such as *shouting*, *interrupting*, *contd.*, *pause*, and *O.S.* (off-screen). Screenwriters usually enclose expressions in parentheses. Expressions can appear alongside speakers and utterances on the same line. We classify such lines into the latter class (speaker or utterance) to avoid ambiguity. **Transition** lines detail the camera movement when the scene changes on screen. Fig 2a shows an example of a well-formatted and parsed screenplay.

3.2 Screenplay Document Issues

Screenplay documents retrieved from online resources like IMSDB [3] and DailyScript [4], or even

[2] Our experiments show that a 48 GB GPU can run inference on a document of length at most 20K tokens

[3] https://imsdb.com/

[4] https://www.dailyscript.com/

EXT. BARTON HOME - DAY	**Slugline**
CLOSE ON: A HOUSE-ARREST ANKLE BRACELET.	**Transition**
CLINT BARTON (O.S.)	**Speaker**
Okay, you see where you're going? Let's work on how to get there.	**Utterance**
Pan up to find...CLINT BARTON, with his daughter, LILA, coaching her as she notches an arrow in her bow.	**Action**
CLINT BARTON	**Speaker**
(CONTD.)	**Expression**
Okay, good...tip down...bow arm out...three fingers-	**Utterance**
LILA BARTON	**Speaker**
Why three?	**Utterance**
CLINT BARTON	**Speaker**
'Cause two's not enough and four's too much-	**Utterance**

(a) Well-formatted parsed screenplay

BARTON HOME - DAY	**Missing scene keyword**
AVENGERS	**Watermark**
CLOSE ON: A HOUSE-ARREST ANKLE BRACELET.	
CL**INT** BARTON (O.S.)	**Name contains keyword**
Okay, you see where you're going? Let's work on how to get there.	
Pan up to find...CLINT BARTON, with his daughter, LILA, coaching her as she notches an arrow in her bow.	
clint barton	**Lowercased Speaker**
ENDGAME	**Watermark**
Okay, good...tip down...bow arm out...three fingers-	
LILA BARTON	**No Whitespace**
Why three?	
CL**INT** BARTON	**Name contains keyword**
** 'Cause two's not enough and four's too much pg 1	**Extra Symbols**

(b) Screenplay with formatting issues

Figure 2: Structural types and formatting issues of screenplay documents (*Avengers Endgame, 2019*)

shooting drafts shared by movie studios, can contain optical character recognition (OCR) errors or disregard formatting conventions. The most common issues found are:

1. **No Whitespace** - The screenwriting convention is that sluglines and action lines should have the smallest indent, followed by utterances and speakers. Screenwriters should separate segments by at least one blank line (Riley, 2009). Non-adherence to these rules or OCR errors might remove all indents and blank lines, making it challenging to determine segment boundaries.

2. **Missing Scene Keywords** - Sluglines omit the *INT* or *EXT* keyword.

3. **Uncapitalization** - Sluglines and speaker names are written in lowercase.

4. **Watermark** - Some screenplays might only be publicly-available as PDF files containing watermark logos or stamps to credit the creator. OCR conversion of the PDF might retain text-based watermarks as spurious letters in screenplay lines.

5. **Speaker Name contains Keyword** - Speaker names might include keywords used in sluglines or transitions, for example, *CLINT*, *CUTTER*, *DEXTER*. These names can confuse rule-based parsers relying on keyword lookup for classification.

6. **Extra Expressions** - Expressions might be misplaced and appear between action lines, instead of with utterances.

7. **Extra Symbols** - Asterisks or page numbers can occur in the beginning or end of screenplay lines.

Fig 2b shows some screenplay formatting issues. The performance of rule-based parsers declines with the preponderance of these anomalies.

3.3 Data Augmentation

Following Agarwal et al. (2014), we create synthetic data to train a robust parser. We collect clean screenplay excerpts annotated line-wise with structural labels. Then, for each issue described in Sec 3.2, we create a *noisy* copy of the labeled data wherein some lines are changed or inserted to contain the corresponding anomaly.

For example, for the **Missing Scene Keyword** type, we remove *INT* and *EXT* keywords from all sluglines. For the **Watermark** type, we randomly insert a new *watermark* line containing spurious English letters. We create the other copies similarly. We augment the original clean corpus with these *noisy* copies and use the combined collection for training and validating our screenplay parser.

3.4 Screenplay Parsing Model

In a typical screenplay, action lines follow sluglines, utterances follow speakers, and transitions mainly occur before sluglines. We model our screenplay parser as a recurrent neural network (RNN) to exploit this structural arrangement.

We encode each line using a concatenation of sentence embeddings and syntactic, orthographic, and keyword-based features. Sentence embeddings capture the semantics of a sentence in a fixed di-

mensional vector representation. Syntactic features include counts of part-of-speech and named entity tags. Orthographic features comprise counts of left and right parentheses and capitalized words. Keyword-based features contain tallies of slugline and transition keywords.

We input the feature representations to a bidirectional RNN to obtain a sequence of hidden vectors. Each hidden vector corresponds to a screenplay line. We feed each vector into a densely connected feed-forward neural network. We compute the label probability as a softmax function over the output neurons of the dense layer. We train the model using class-weighted cross entropy loss.

4 Screenplay Character Coreference Resolution

4.1 Problem Setup

Character coreference resolution is a document-level hard-clustering[5] task where each cluster is a set of text spans that refer to a single unique character. We call text spans that refer to some character as character mentions. Character mentions can occur in any structural segment of the screenplay.

4.2 Screenplay Coreference Model

We adapt the word-level coreference resolution model of Dobrovolskii (2021) to the screenplay character coreference resolution task. The model first finds the coreference links between individual words and then expands each word to the subsuming span. We chose this model because it achieved near-SOTA[6] performance on the OntoNotes dataset while having a simple architecture and maintaining quadratic complexity.

Following Lee et al. (2017), we formulate the coreference resolution task as a set of antecedent assignments y_i for each word i. Possible y_i values are $\mathcal{Y}_i = \{\epsilon, 1, \ldots i-1\}$. $y_i = \epsilon$ if word i does not have an antecedent or is not a character mention. We model the probability distribution $P(y_i)$ over candidate antecedents as:

$$P(y_i) = \frac{e^{s(i,y_i)}}{\sum_{y' \in \mathcal{Y}_i} e^{s(i,y')}} \quad (1)$$

$s(i,j)$ is the coreference score between words i and j. We fix $s(i,\epsilon) = 0$. Following steps show how we compute $s(i,j)$.

[5] a text span can belong to at most one cluster

[6] Dobrovolskii (2021) achieves 81.0 F1. SOTA is 83.3 F1

Word Representations. Given a screenplay $\mathcal{S}$ containing n words, we tokenize each word to obtain m wordpiece subtokens. We encode the subtokens using BERT-based transformers (Devlin et al., 2019) to obtain contextualized subtoken embeddings $\mathbf{T}$. We do not pass the whole subtoken sequence to the transformer because the sequence length m is usually greater than the transformer's context window. Instead, we split the subtoken sequence into non-overlapping segments and encode each segment separately. Joshi et al. (2019) showed that overlapping segments provided no improvement. We obtain the word representations $\mathbf{X}$ as a weighted sum of its subtoken embeddings. We find the weights by applying a softmax function over the attention scores of its subtokens. We calculate the subtoken attention scores $\mathbf{A}$ by a linear transformation $\mathbf{W}$ on $\mathbf{T}$.

$$\mathbf{A} = \mathbf{T} \cdot \mathbf{W} \quad (2)$$

Character Scores. The character score of a word calculates its likelihood to be the head word of a character mention. We calculate character scores because we only want to model the coreference between characters instead of all entity types. We obtain word-level character representations $\mathbf{Z}$ by concatenating the word representations $\mathbf{X}$ with word feature embeddings. The word features include part-of-speech, named entity, and structural tags of the word. The word's structural tag is the structural tag of the screenplay line containing the word. We apply a bidirectional RNN to $\mathbf{Z}$ to obtain hidden vectors $\mathbf{H}$. We input each hidden vector $\mathbf{H_i}$ to a feed-forward neural network with a single output neuron to find the character score $s_r(i)$ for word i.

$$\mathbf{H} = \mathbf{RNN_r}(\mathbf{Z}) \quad (3)$$

$$s_r(i) = \mathbf{FFN_r}(\mathbf{H_i}) \quad (4)$$

Coarse Coreference Scores. The coarse coreference score is a computationally efficient but crude estimate of how likely two words corefer each other. We calculate it as the sum of the bilinear transformation $\mathbf{W_c}$ of the word embeddings and the character scores of the words.

$$s_c(i,j) = \mathbf{X_i} \cdot \mathbf{W_c} \cdot \mathbf{X_j^\intercal} + s_r(i) + s_r(j) \quad (5)$$

Antecedent Coreference Score. We retain the top k likely antecedents of each word according to the coarse coreference scores s_c. We encode a word and candidate antecedent pair (i,j) by concatenating their word embeddings, the element-wise product of their word embeddings, and a pairwise representation ϕ which encodes the distance between

words i and j, and whether they are spoken by the same character. We feed the word-antecedent representations to a feed-forward neural network to obtain antecedent coreference scores $s_a(i, j)$.

$$s_a(i, j) = \mathbf{FFN_a}([\mathbf{X_i}; \mathbf{X_j}; \mathbf{X_i} \odot \mathbf{X_j}; \phi]) \quad (6)$$

The final coreference score of the word and candidate antecedent pair (i, j) is the sum of coarse and antecedent coreference scores $s(i, j) = s_c(i, j) + s_a(i, j)$. The predicted antecedent for word i is the antecedent with the maximum $s(i, j)$ score. Word i has no antecedents if $s(i, j)$ is negative for all k candidate antecedents.

Span Boundary Detection. We find the span boundaries of the words that are coreferent with some other word. We concatenate the word embedding with embeddings of neighboring words and pass the pairwise representations through a convolutional neural network followed by a feed-forward network to get start and end scores. The preceding and succeeding words with maximum start and end scores mark the span boundaries.

We obtain the final coreference clusters by using graph traversal on the predicted antecedent relationship between the head words of the spans. The time and space complexity of the model is $O(n^2)$.

4.3 Training

The large document size of screenplays prevents us from calculating gradients from the entire document within our resource constraints. Therefore, we split the screenplay at segment boundaries (segments are defined in sec 3.1) into non-overlapping subdocuments and train on each subdocument separately. We do not split at scene boundaries because scenes can get very long.

Following Dobrovolskii (2021), we use marginal log-likelihood $\mathcal{L}_{\text{MLL}}$ and binary cross entropy $\mathcal{L}_{\text{BCE}}$ to train the coarse and antecedent coreference scorers. We optimize the marginal likelihood because the antecedents are latent and only the clustering information is available, as shown in Eq 7. $\mathcal{G}_i$ denotes the set of words in the gold cluster containing word i. The binary cross entropy term $\mathcal{L}_{\text{BCE}}$ improves the coreference scores for individual coreferring word pairs. We scale it by a factor α. We use cross entropy loss to train the character scorer and span boundary detection modules, denoted as $\mathcal{L}_{\text{Char}}$ and $\mathcal{L}_{\text{Span}}$, respectively. We train the coreference, character scorer and span boundary detection modules jointly (Eq 8).

$$\mathcal{L}_{\text{MLL}} = -\log \prod_{i=1}^{n} \sum_{\hat{y} \in \mathcal{Y}_i \cap \mathcal{G}_i} P(\hat{y}) \quad (7)$$

$$\mathcal{L} = \mathcal{L}_{\text{MLL}} + \alpha \mathcal{L}_{\text{BCE}} + \mathcal{L}_{\text{Char}} + \mathcal{L}_{\text{Span}} \quad (8)$$

4.4 Inference

Unlike training, we cannot run inference separately on non-overlapping subdocuments because we will miss coreference links between words occuring in different subdocuments and each coreference cluster will be confined to a single subdocument. We devise two approaches to scale inference to long screenplays, one based on fusing coreference scores and the other is a hierarchical method.

4.4.1 Fusion-Based Inference

We split the screenplay into *overlapping* subdocuments and run inference separately on each to obtain coreference scores $s_k(i, j)$ for each subdocument k. If a word pair (i, j) lies within the overlap region of two adjacent subdocuments k_1 and k_2, we might calculate two different coreference scores $s_{k_1}(i, j)$ and $s_{k_2}(i, j)$. We average the two scores to obtain the final coreference value $s(i, j)$ and use it to find the final coreference clusters. This method finds coreference scores for all word pairs whose separation is less than the overlap length.

4.4.2 Hierarchical Inference

We split the screenplay into *non-overlapping* subdocuments. We run inference and find coreference clusters for each subdocument separately. For each subdocument coreference cluster, we sample some *representative* words that have the highest character scores s_r. We calculate the coarse and antecedent coreference scores for every word pair (i, j), where words i and j are representative words of coreference clusters from different subdocuments. If the average coreference score $s(i, j)$ is positive, we merge the corresponding subdocument coreference clusters. We obtain the final coreference clusters after no further merging can take place. This method allows merging distant clusters together.

5 Experiments

5.1 Screenplay Parsing

Dataset. We use the screenplay parsing dataset of Baruah et al. (2021). We inject noise into this dataset according to Sec 3.3 and create the *MovieParse* dataset. The *MovieParse* dataset contains clean and noisy versions of 39 screenplay excerpts, totaling 224,260 labeled lines.

Baseline. We employ the rule-based parser of Baruah et al. (2021) as our baseline.

Implementation. We use the pretrained Sentence MPNet (Masked and Permuted Language Modeling) transformer (Song et al., 2020) to get sentence embeddings. The model uses Siamese and triplet network structure to obtain sentence representations. We employ English Spacy models (Honnibal et al., 2020) to find the syntactic features. The parser's RNN layer is a single layer LSTM with 256-dimensional hidden vectors. We train the parser on sequences of 10 screenplay lines. We use learning rates of 1e-3 and 1e-5 for the sentence encoder and LSTM, respectively. We train for 5 epochs using Adam optimizer.

Evaluation. We use leave-one-movie-out cross-validation and average the performance across the 39 excerpts to obtain the final evaluation scores. We use per-class F1 as the evaluation metric.

5.2 Screenplay Coreference Resolution

Movie	W	M	C
Avengers Endgame (*f*)	35,816	5,025	71
Dead Poets Society (*f*)	26,200	3,778	51
John Wick (*f*)	24,954	2,580	34
Prestige (*f*)	35,910	5,140	34
Quiet Place (*f*)	27,843	2,786	9
Zootopia (*f*)	27,127	3,677	113
Shawshank Redemption (*e*)	8,090	888	44
The Bourne Identity (*e*)	8,087	911	39
Inglourious Basterds (*e*)	7,777	1,008	23
Total	201,804	25,793	418

Table 1: Descriptive statistics of the *MovieCoref* dataset. *f* = full-length, *e* = excerpt, *W* = Words, *M* = Mentions, *C* = Characters. The excerpt scripts were annotated by Baruah et al. (2021).

Dataset. We label six full-length screenplays for character coreference using the annotation guidelines of Baruah et al. (2021). The scripts are publicly available from IMSDB. Three trained individuals annotated two unique screenplays each plus an additional script excerpt previously labeled by experts for rater-reliability measures. The average LEA F1 (Moosavi and Strube, 2016) of the annotators against the expert labels is 85.6. We used the CorefAnnotator tool to annotate the screenplay documents (Reiter, 2018).

The six movies are Avengers Endgame, 2019; Dead Poets Society, 1989; John Wick, 2014; Prestige, 2006; Quiet Place, 2018; and Zootopia, 2016. We add these movies to the coreference dataset annotated by Baruah et al. (2021) to create the *MovieCoref* dataset. The average document length of the full-length screenplays is about 30K words. *MovieCoref* covers 25,793 character mentions for 418 characters in 201,804 words. The maximum active entity count (defined in sec 1) is 54. The same statistic for OntoNotes and LitBank is 24 and 18 respectively (Toshniwal et al., 2020). Table 1 shows per-movie statistics of *MovieCoref* dataset.

Baseline. We use the screenplay coreference model of Baruah et al. (2021) as our baseline. It combines the neural model of Lee et al. (2018) with structural rules to adapt to the movie domain.

Implementation. We retain the architecture of the word-level coreference model of Dobrovolskii (2021) for the word encoder, coreference scorers, and span boundary detection modules. We pretrain these modules on the OntoNotes corpus. Following Dobrovolskii (2021), we use RoBERTa (Zhuang et al., 2021) to encode the subtokens. The character scorer uses a single-layer bidirectional GRU with 256-dimensional hidden vectors.

We train the coreference model using a learning rate of 2e-5 for the RoBERTa transformer and 2e-4 for the other modules. We decrease the learning rates linearly after an initial warmup of 50 training steps. We use L2 regularization with a decay rate of 1e-3. The size of the training subdocuments is 5120 words because it is the maximum we could fit in 48 GB A40 NVIDIA GPUs. We retain the top 50 antecedent candidates during the pruning stage. We set the binary-cross-entropy scaling factor $\alpha = 0.5$. Appendix A contains additional details.

Evaluation. We use leave-one-movie-out cross-validation to evaluate the model. We obtain the final evaluation score by averaging across the six full-length screenplays. The conventional evaluation metric for coreference resolution is CoNLL F1, which is the average of the F1 scores of three metrics: MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), and $CEAF_e$ (Luo, 2005). Moosavi and Strube (2016) pointed out interpretability and discriminative issues with these metrics and proposed the alternative LEA (Link-Based Entity Aware) metric. LEA calculates the weighted sum of the resolution scores of the clusters. The resolution score of a cluster is the fraction of detected coreference links, and its weight equals its cardinality. We use LEA F1 as the primary evaluation metric.

Issue	Slugline		Action		Speaker		Utterance		Expression		Transition	
	B	R	B	R	B	R	B	R	B	R	B	R
-	**97.8**	95.8	89.8	**98.8**	97.8	**99.6**	93.5	**99.3**	86.2	**98.1**	89.4	**97.8**
No Whitespace	0.6	**92.3**	0.4	**97.5**	92.9	**98.9**	66.3	**98.3**	81.1	**98.7**	2.9	**95.8**
Missing Scene Kw	64.2	**95.6**	87.4	**98.8**	97.8	**99.6**	93.5	**99.4**	86.2	**98.1**	89.4	**98.3**
Uncap Slugline	64.9	**95.5**	87.5	**98.8**	97.8	**99.6**	93.5	**99.4**	86.2	**98.1**	89.4	**97.8**
Uncap Speaker Name	**97.8**	95.7	63.8	**98.7**	64.7	**99.5**	64.5	**99.3**	60.4	**97.7**	88.3	**98.3**
Watermark	65.0	**94.3**	36.8	**96.8**	41.8	**90.8**	63.7	**97.9**	74.8	**90.9**	57.6	**93.1**
Speaker contains Kw	**97.8**	95.6	74.4	**98.8**	72.6	**99.6**	72.7	**99.4**	66.0	**98.2**	14.9	**98.1**
Extra Expressions	**97.8**	95.5	88.4	**98.4**	97.8	**99.0**	93.8	**98.1**	97.8	**98.6**	89.4	**98.1**
Extra Symbols	89.7	**91.4**	64.9	**96.2**	**94.8**	91.2	75.1	**97.2**	83.2	**94.3**	78.4	**93.7**

Table 2: Per-class F1 scores of rule-based parser Baruah et al. (2021) (B) and our parser (R) across different formatting issues on the *MovieParse* dataset. *Kw* = Keyword, *Uncap* = Uncapitalized.

Model	MUC			B^3			$CEAF_e$			LEA			CoNLL
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	F1
Baseline	89.3	79.2	83.7	54.8	63.3	57.5	**61.5**	**40.9**	**47.9**	63.2	54.7	57.3	63.1
Fusion	**93.9**	**92.8**	**93.3**	**81.3**	**69.0**	**74.5**	34.7	62.9	43.3	**81.0**	**68.7**	**74.2**	70.4
Hierarchical	94.2	92.4	93.2	70.6	78.8	73.8	39.5	60.4	46.2	70.3	78.6	73.5	**71.1**

Table 3: Leave-one-movie-out cross-validation scores on the character coreference resolution task on the *MovieCoref* dataset. Baseline refers to the screenplay coreference model of Baruah et al. (2021) built on top Lee et al. (2018).

6 Results

6.1 Screenplay Parsing

Table 2 compares the per-class F1 scores of the rule-based parser of Baruah et al. (2021) against our parser for different formatting issues. Our parser performs significantly better across most structural tags and formatting variations of the screenplay document (*t*-test, $p < 1e^{-4}$). The rule-based parser is a better slugline detector for well-formatted screenplays, but its performance degrades significantly for *noisy* screenplays. The performance of our parser varies significantly less than the rule-based parser across document issues, proving its robustness (*F*-test, $p < 1e^{-4}$).

6.2 Screenplay Coreference Resolution

Table 3 shows the cross-validation results of our model and the baseline (Baruah et al., 2021) on the character coreference resolution task. For the fusion-based inference, we split the screenplay into overlapping subdocuments, each of length 5120 words with an overlap of 2048 words. For the hierarchical inference, we split the screenplay into non-overlapping subdocuments, each of length 5120 words, and sample three representative words per cluster. Both inference approaches achieved significantly better F1 scores than the baseline except on the $CEAF_e$ metric but did not differ significantly from each other (*t*-test, $p < 0.05$). The hierarchical approach retrieves more coreference links (+10 LEA recall) but is less precise (-11 LEA precision) than the fusion-based approach. This might be because the hierarchical approach performs a second round of coreference clustering, which merges distant clusters but also introduces wrong coreference links. We use Bonferroni correction (n = 3) to adjust for multiple comparisons.

6.2.1 Character Scorer Ablation

Model	LEA F1
(w/o character scorer)	72.2
(w/ character scorer)	**74.2**

Table 4: LEA F1 score of fusion-based inference with and without the character scorer module

The main difference between our model and Dobrovolskii's (2021) word-level coreference model is the inclusion of the character scorer module. Excluding the character scorer module implies that we do not have the s_r terms in Eq 5 and the $\mathcal{L}_{\text{Char}}$ term in Eq 8. Table 4 shows that the coreference perfor-

mance of the fusion-based approach improves (+2 LEA F1) by adding the character scorer module. Similar results hold for the hierarchical approach.

6.2.2 Fusion-Based Inference

SubDoc (words)	Overlap (words) 256	512	1024	2048
2048	51.5 (7.6)	53.1 (7.6)	-	-
3072	52.4 (7.7)	59.2 (7.7)	63.8 (7.7)	-
4096	58.6 (7.8)	58.4 (7.8)	68.7 (7.8)	-
5120	60.1 (7.9)	62.4 (7.9)	70.8 (8.0)	74.2 (8.0)
8192	64.6 (8.5)	67.0 (8.5)	72.9 (8.5)	76.6 (8.5)

Table 5: LEA F1 scores of fusion-based inference for different subdocument (SubDoc) and overlap sizes. Parenthesized numbers are GPU memory usage in GB.

Table 5 shows the coreference performance and memory usage of the fusion-based inference approach across different subdocument and overlap lengths. Performance improves significantly if we split the screenplay into larger subdocuments with greater overlap (t-test, $p < 0.05$). Increasing the subdocument size enables the model to directly find the coreference score of more word pairs. Increasing the overlap between adjacent subdocuments allows the model to score all word pairs whose separation is less than the overlap length. Memory consumption remains almost steady for increasing overlap sizes at a given subdocument length.

6.2.3 Hierarchical Inference

SubDoc (words)	Representative Words 1	2	3	4
2048	61.7 (9.2)	67.5 (14.0)	-	-
3072	63.6 (8.8)	72.1 (12.6)	73.5 (18.8)	-
4096	69.8 (8.6)	73.0 (12.0)	74.9 (17.4)	-
5120	72.9 (8.5)	72.3 (11.3)	73.5 (15.8)	-
8192	72.9 (8.7)	75.1 (10.6)	75.6 (14.3)	75.9(19.5)

Table 6: LEA F1 scores of hierarchical inference for different subdocument sizes (SubDoc) and number of representative mentions. Parenthesized numbers are GPU memory usage in GB.

Table 6 shows the coreference performance and memory usage of the hierarchical inference approach across different subdocument sizes and number of representative words. Similar to the fusion-based approach, performance improves significantly upon increasing the subdocument length (t-test, $p < 0.05$). Sampling more words per subdocument cluster also improves performance because they provide more information from different discourse locations about the character. However, it substantially increases memory usage. Memory consumption decreases for greater subdocument sizes for a given number of representative words. This might be because increasing the subdocument length decreases the total number of subdocuments, which reduces the number of clusters obtained from the first round of coreference scoring.

7 Conclusion

We introduce two movie screenplay datasets for parsing and character coreference resolution tasks. We develop a robust screenplay parser that can handle various document formatting issues. We devise inference approaches to scale coreference models to long documents without drastically increasing memory consumption and evaluate them on full-length screenplays. Future work entails applying our screenplay coreference model to gather longitudinal insights on movie character interactions.

8 Limitations

The coreference annotations of the *MovieCoref* dataset exclude plural character mentions because the annotation guidelines did not cover them (Baruah et al., 2021). It contains few singleton coreference clusters (65). Our model only identifies singular characters and cannot retrieve singleton clusters. All the movies in the dataset have a linear narrative. Non-linear stories can confuse a coreference model because of time skips and flashbacks which is not explored in our work. Both our inference approaches require at least 10 GB of GPU memory for finding coreference clusters from full-length screenplays.

9 Ethics Statement

Our work adheres to the ACL Ethics Policy. Using our proposed models, we can scale coreference resolution to long documents while leveraging transformer-based mention-pair scorers and without substantially increasing memory consumption.

References

Apoorv Agarwal, Sriramkumar Balasubramanian, Jiehan Zheng, and Sarthak Dash. 2014. Parsing screenplays for extracting social networks from movies. In *Proceedings of the 3rd Workshop on Computational Linguistics for Literature (CLFL)*, pages

50–58, Gothenburg, Sweden. Association for Computational Linguistics.

Paul Argentini. 1998. *Elements of Style for Screen Writers*. Lone Eagle Publishing Company.

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *The first international conference on language resources and evaluation workshop on linguistics coreference*, volume 1, pages 563–566. Citeseer.

David Bamman, Olivia Lewke, and Anya Mansoor. 2020. An annotated dataset of coreference in English literature. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 44–54, Marseille, France. European Language Resources Association.

David Bamman, Brendan O'Connor, and Noah A. Smith. 2013. Learning latent personas of film characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 352–361, Sofia, Bulgaria. Association for Computational Linguistics.

David Bamman, Sejal Popat, and Sheng Shen. 2019. An annotated dataset of literary entities. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2138–2144, Minneapolis, Minnesota. Association for Computational Linguistics.

Sabyasachee Baruah, Sandeep Nallan Chakravarthula, and Shrikanth Narayanan. 2021. Annotation and evaluation of coreference resolution in screenplays. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2004–2010, Online. Association for Computational Linguistics.

Bernd Bohnet, Chris Alberti, and Michael Collins. 2022. Coreference resolution through a seq2seq transition-based system.

Mariona Coll Ardanuy and Caroline Sporleder. 2014. Structure-based clustering of novels. In *Proceedings of the 3rd Workshop on Computational Linguistics for Literature (CLFL)*, pages 31–39, Gothenburg, Sweden. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. 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.

Micha Elsner. 2012. Character-based kernels for novelistic plot structure. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 634–644, Avignon, France. Association for Computational Linguistics.

David Elson, Nicholas Dames, and Kathleen McKeown. 2010. Extracting social networks from literary fiction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 138–147, Uppsala, Sweden. Association for Computational Linguistics.

Philip John Gorinski and Mirella Lapata. 2015. Movie script summarization as graph-based scene extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1066–1076, Denver, Colorado. Association for Computational Linguistics.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spacy: Industrial-strength natural language processing in python.

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.

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.

Vincent Labatut and Xavier Bost. 2019. Extraction and analysis of fictional character networks. *ACM Computing Surveys*, 52(5):1–40.

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.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 25–32.

Victor R. Martinez, Krishna Somandepalli, and Shrikanth Narayanan. 2022. Boys don't cry (or kiss or dance): A computational linguistic lens into gendered actions in film. *PLOS ONE*, 17(12):1–23.

Nafise Sadat Moosavi and Michael Strube. 2016. Which coreference evaluation metric do you trust? a proposal for a link-based entity aware metric. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 632–642, Berlin, Germany. Association for Computational Linguistics.

Eric T. Nalisnick and Henry S. Baird. 2013. Character-to-character sentiment analysis in shakespeare's plays. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 479–483, Sofia, Bulgaria. Association for Computational Linguistics.

Andrew Piper, Richard Jean So, and David Bamman. 2021. Narrative theory for computational narrative understanding. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 298–311, Online and Punta Cana, Dominican Republic. 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.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2022. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

Anil Ramakrishna, Victor R. Martínez, Nikolaos Malandrakis, Karan Singla, and Shrikanth Narayanan. 2017. Linguistic analysis of differences in portrayal of movie characters. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1669–1678, Vancouver, Canada. Association for Computational Linguistics.

Nils Reiter. 2018. CorefAnnotator - A New Annotation Tool for Entity References. In *Abstracts of EADH: Data in the Digital Humanities*.

Christopher Riley. 2009. *The Hollywood standard: the complete and authoritative guide to script format and style*. Michael Wiese Productions.

Maarten Sap, Marcella Cindy Prasettio, Ari Holtzman, Hannah Rashkin, and Yejin Choi. 2017. Connotation frames of power and agency in modern films. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2329–2334, Copenhagen, Denmark. Association for Computational Linguistics.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. *Advances in Neural Information Processing Systems*, 33:16857–16867.

Shashank Srivastava, Snigdha Chaturvedi, and Tom Mitchell. 2016. Inferring interpersonal relations in narrative summaries.

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.

R. Turetsky and N. Dimitrova. 2004. Screenplay alignment for closed-system speaker identification and analysis of feature films. In *2004 IEEE International Conference on Multimedia and Expo (ICME) (IEEE Cat. No.04TH8763)*, volume 3, pages 1659–1662 Vol.3.

Hardik Vala, David Jurgens, Andrew Piper, and Derek Ruths. 2015. Mr. bennet, his coachman, and the archbishop walk into a bar but only one of them gets recognized: On the difficulty of detecting characters in literary texts. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 769–774, Lisbon, Portugal. Association for Computational Linguistics.

Dieter Van Rijsselbergen, Barbara Van De Keer, Maarten Verwaest, Erik Mannens, and Rik Van de Walle. 2009. Movie script markup language. In *Proceedings of the 9th ACM Symposium on Document Engineering*, DocEng '09, page 161–170, New York, NY, USA. Association for Computing Machinery.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Sixth Message Understanding Conference (MUC-6): Proceedings of a Conference Held in Columbia, Maryland, November 6-8, 1995*.

David R Winer and R Michael Young. 2017. Automated screenplay annotation for extracting storytelling knowledge. In *Thirteenth Artificial Intelligence and Interactive Digital Entertainment Conference*.

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.

Michael Yoder, Sopan Khosla, Qinlan Shen, Aakanksha Naik, Huiming Jin, Hariharan Muralidharan, and Carolyn Rosé. 2021. FanfictionNLP: A text processing pipeline for fanfiction. In *Proceedings of the Third Workshop on Narrative Understanding*, pages 13–23, Virtual. Association for Computational Linguistics.

Mo Yu, Yisi Sang, Kangsheng Pu, Zekai Wei, Han Wang, Jing Li, Yue Yu, and Jie Zhou. 2022. Few-shot character understanding in movies as an assessment to meta-learning of theory-of-mind. *arXiv preprint arXiv:2211.04684*.

Liu Zhuang, Lin Wayne, Shi Ya, and Zhao Jun. 2021. A robustly optimized BERT pre-training approach with post-training. In *Proceedings of the 20th Chinese National Conference on Computational Linguistics*, pages 1218–1227, Huhhot, China. Chinese Information Processing Society of China.

A Appendix

Implementation Details : For the cross-validation experiments on the *MovieCoref* dataset, we update the model parameters after every forward pass on a subdocument. Within the forward pass, we use batching for the character scoring, antecedent coreference scoring, and span boundary detection stages to decrease GPU memory consumption. We use a batch size of 64. We feed word sequences containing 256 words to the character scorer RNN. The pairwise feature encoder ϕ also encodes the document genre. OntoNotes defines seven genres such as newswire, broadcast, and telephone conversations. We set the genre as web data. Performance did not differ significantly from the other genres. Baruah et al. (2021) suggested inserting the expression *says* between speaker names and utterances before finding coreference clusters in screenplays. We forgo this preprocessing step because we did not observe any performance improvement.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section 8

☑ A2. Did you discuss any potential risks of your work?
Section 9

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☒ Did you use or create scientific artifacts?

Left blank.

☐ B1. Did you cite the creators of artifacts you used?
No response.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
No response.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
No response.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
No response.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
No response.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
No response.

C ☑ Did you run computational experiments?

Section 5

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 5

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 5

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 6

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 5

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.