Hierarchical Encoders for Modeling and Interpreting Screenplays

Gayatri Bhat* **Avneesh Saluja** **Melody Dye** **Jan Florjanczyk**
Bloomberg Netflix
New York, NY, USA Los Angeles, CA, USA
gbhat7@bloomberg.net {asaluja,mdye,jflorjanczyk}@netflix.com

Abstract

While natural language understanding of long-form documents remains an open challenge, such documents often contain structural information that can inform the design of models encoding them. Movie scripts are an example of such richly structured text – scripts are segmented into scenes, which decompose into dialogue and descriptive components. In this work, we propose a neural architecture to encode this structure, which performs robustly on two multi-label tag classification tasks without using handcrafted features. We add a layer of insight by augmenting the encoder with an unsupervised 'interpretability' module, which can be used to extract and visualize narrative trajectories. Though this work specifically tackles screenplays, we discuss how the underlying approach can be generalized to a range of structured documents.

1 Introduction

As natural language understanding of sentences and short documents continues to improve, interest in tackling longer-form documents such as academic papers (Ren et al., 2014; Bhagavatula et al., 2018), novels (Iyyer et al., 2016) and screenplays (Gorinski and Lapata, 2018) has been growing. Analyses of such documents can take place at multiple levels, e.g. identifying both document-level labels (such as genre) and narrative trajectories (how do levels of humor and romance vary over the course of a romantic comedy?). However, one key challenge for these tasks is the low signal-to-noise ratio in lengthy texts (as indicated by the performance of such models on curated datasets like NarrativeQA (Kočiský et al., 2018)), which makes it difficult to apply end-to-end (E2E) neural network solutions that have recently achieved state-of-the-art on other tasks (Barrault et al., 2019; Williams et al., 2018; Wang et al., 2019).

Instead, models either rely on a) a *pipeline* that provides a battery of syntactic and semantic information from which to craft features (e.g., the BookNLP pipeline (Bamman et al., 2014) for literary text, graph-based features (Gorinski and Lapata, 2015) for movie scripts, or outputs from a discourse parser (Ji and Smith, 2017) for text categorization) and/or b) the *linguistic intuitions* of the model designer to select features relevant to the task at hand (e.g., rather than ingest the entire text, Bhagavatula et al. (2018) only consider certain sections like the title and abstract of an academic publication). While there is much to recommend these approaches, E2E neural modeling offers several key advantages: it obviates the need for auxiliary feature-generating models, minimizes the risk of error propagation, and offers improved generalization across large-scale corpora. This work explores how the inherent structure of a document class can be leveraged to facilitate an E2E approach. We focus on screenplays, investigating whether we can effectively extract key information by first segmenting them into scenes, and further exploiting the structural regularities within each scene.

With an average of >20k tokens per script in our evaluation corpus, extracting salient aspects is far from trivial. Through a series of carefully controlled experiments, we show that a structure-aware approach significantly improves document classification by effectively collating sparsely distributed information. Further, this method produces both document- and scene-level embeddings, which can be used downstream to visualize narrative trajectories of interest (e.g., the prominence of various themes across a script). The overarching strategy of this work is to incorporate structural priors as biases into the neural *architecture* itself (e.g., Socher et al. (2013), Strubell et al. (2018), *inter alia*), whereby, as Henderson (2020) observe, "locality in the model structure can reflect locality in the linguistic structure" to boost

*Work done during an internship at Netflix

Proceedings of the 3rd Workshop on Narrative Understanding, pages 1–12
June 11, 2021.

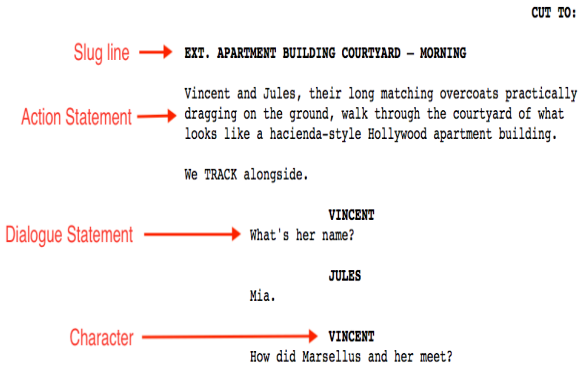

Figure 1: A portion of the screenplay for *Pulp Fiction*, annotated with the common scene components.

accuracy over feature-engineering approaches. The methods we propose can readily generalize to any long-form text with an exploitable internal structure, including novels (chapters), theatrical plays (scenes), chat logs (turn-taking), online games (levels/rounds/gameplay events), and academic texts (sections and subsections).

We begin by detailing how a script can be formally decomposed first into scenes and further into granular elements with distinct discourse functions, in §2. We then propose an encoder based on hierarchical attention (Yang et al., 2016) that effectively leverages this structure in §3. In §5.3, the predictive performance of the hierarchical encoder is validated on two multi-label tag prediction tasks, one of which rigorously establishes the utility of modeling structure at multiple granularities (i.e. at the level of line, scene, and script). Notably, while the resulting scene-encoded representation is useful for prediction tasks, it is not amenable to easy interpretation or examination. To shed light on the encoded document representations, in §4, we propose an unsupervised interpretability module that can be attached to an encoder of any complexity. §5.5 outlines our application of this module to the scene encoder, and the resulting visualizations of the screenplay, which illustrate how plot elements vary over the course of the narrative arc. §6 draws connections to related work, before concluding.

2 Script Structure

Movie and television scripts (or screenplays) are traditionally segmented into *scenes*, with a rough rule of thumb being that each scene lasts about a minute on-screen. A scene is not necessarily a distinct narrative unit (which is most often a sequence of several consecutive scenes), but is constituted by a piece of continuous action at a single location.

Title	Line	Scene	Type	Character	Text
Pulp Fiction	204	4	Scene		EXT. APART..
Pulp Fiction	205	4	Action		Vincent and Jules.
Pulp Fiction	206	4	Action		We TRACK...
Pulp Fiction	207	4	Dial.	VINCENT	What's her name?
Pulp Fiction	208	4	Dial.	JULES	Mia.
Pulp Fiction	209	4	Dial.	VINCENT	How did...

Table 1: Post-processed version of Fig.1.

Fig. 1 contains a segment of a scene from the screenplay for the *Pulp Fiction*, a 1994 American film. These segments tend to follow a standard format. Each scene starts with a scene heading or 'slug line' that briefly describes the scene setting. A sequence of statements follow, and screenwriters typically use formatting to distinguish between dialogue and action statements (Argentini, 1998). A dialogue identifies the character who utters it either on- or off-screen (the latter is often indicated with '(V.O.)' for voice-over). Parentheticals might be used to include special instructions regarding dialogue delivery. Action statements are all non-dialogue constituents of the screenplay "often used by the screenwriter to describe character actions, camera movement, appearance, and other details" (Pavel et al., 2015). In this work, we consider action and dialogue statements, as well as character identities for each dialogue segment, ignoring slug lines and parentheticals.

3 Hierarchical Scene Encoders

The large size of a movie script makes it computationally infeasible for recurrent encoders to ingest these screenplays as single blocks of text. Instead, we propose a hierarchical encoder that mirrors the structure of a screenplay (§2) – a sequence of scenes, each of which is an interwoven sequence of action and dialogue statements. The encoder is three-tiered, as illustrated in Fig. 2, and processes the text of a script as follows.

3.1 Model Architecture

First, an **action-statement encoder** transforms the sequence of words in an action statement (represented by their pretrained word embeddings) into an action statement embedding. Next, an **action-scene encoder** transforms the chronological sequence of action statement embeddings within a scene into an action scene embedding. Analogously, a **dialogue-statement encoder** and a **dialogue-scene encoder** generate dialogue statement embeddings and aggregate them into dialogue

scene embeddings. To incorporate character information, characters are represented as embeddings (randomly initialized and updated during model training), and an average of embeddings of all characters with at least one dialogue in the scene is computed.[1] Finally, the action, dialogue and averaged character embeddings for a scene are concatenated into a single scene embedding. Scene-level predictions can be obtained by feeding scene embeddings into a subsequent neural module, e.g. a feedforward layer for supervised tagging. Alternatively, a final **script encoder** can be used to transform the sequence of scene embeddings into a script embedding representing the entire screenplay.

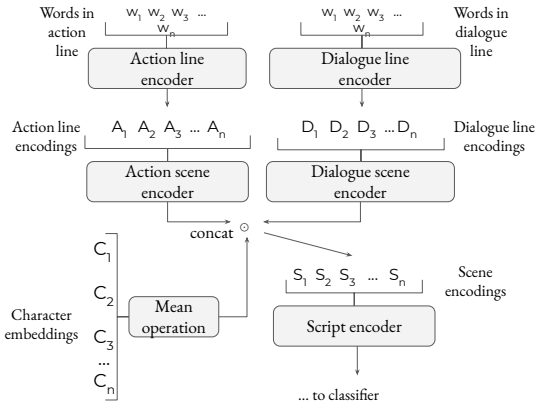

Figure 2: The architecture of our script encoder, largely following the structure in Fig. 1.

A key assumption underlying the model is that action and dialogue statements – as instances of written narrative and spoken language respectively – are distinct categories of text that must be processed separately. We evaluate this assumption in §5.3.

3.2 Encoders

The proposed model incorporates strong inductive biases regarding the overall structure of input documents. In addition, since the aforementioned encoders §3.1 are underspecified, we evaluate three instantiations of the encoder components:

1. **Sequential (GRU):** A bidirectional GRU (Bahdanau et al., 2015) encodes input sequences (of words, statements or scenes). Given a sequence of input embeddings $\mathbf{e}_1, \ldots, \mathbf{e}_T$, we obtain GRU outputs $\mathbf{c}_1, \ldots, \mathbf{c}_T$, and use $\mathbf{c}_T$ as the recurrent encoder's final output.

2. **Sequential with Attention (GRU + Attn):** Attention (Bahdanau et al., 2015) is used to combine $\mathbf{c}_1, \ldots, \mathbf{c}_T$. This allows more or less informative inputs to be filtered accordingly. We calculate attention weights using a parametrized vector $\mathbf{p}$ of the same dimensionality as the GRU outputs (Sukhbaatar et al., 2015; Yang et al., 2016):

$$\alpha_i = \frac{\mathbf{p}^T \mathbf{c}_i}{\Sigma_{j=1}^{T} \mathbf{p}^T \mathbf{c}_j} \quad (1)$$

These weights are used to compute the final encoder output:

$$\mathbf{c} = \Sigma_{j=1}^{T} \alpha_i \mathbf{c}_i \quad (2)$$

3. **Bag-of-Embeddings with Attention (BoE + Attn):** These encoders disregard sequential information to compute an attention-weighted average of the encoder's inputs:

$$\alpha_i = \frac{\mathbf{p}^T \mathbf{e}_i}{\Sigma_{j=1}^{T} \mathbf{p}^T \mathbf{e}_j} \quad (3)$$

$$\mathbf{c} = \Sigma_{j=1}^{T} \alpha_i \mathbf{e}_i \quad (4)$$

In contrast, a bag-of-embeddings (BoE) encoder computes a simple average of its inputs. While defining a far more constrained function space than recurrent encoders, BoE and BoE + Attn representations have the advantage of remaining in the input word embedding space. We leverage this property in §4 where we develop an interpretability layer on top of the encoder outputs.

3.3 Loss for Tag Classification

The final script embedding is passed into a feedforward classifier (FFNN). As both supervised learning tasks in our evaluation are multi-label classification problems, we use a variant of a simple multi-label one-versus-rest loss, where correlations among tags are ignored. The tag sets have high cardinalities and the fractions of positive samples are inconsistent across tags (see Appendix Tables 3 & 4); this motivates the use of a reweighted loss function:

$$L(y, z) = \frac{1}{NL} \Sigma_{i=1}^{N} \Sigma_{j=1}^{L} [y_{ij} \log \sigma(z_{ij}) + \lambda_j (1 - y_{ij})(1 - \log \sigma(z_{ij}))] \quad (5)$$

where N is the number of samples, L is the number of tag labels, $y \in \{0, 1\}$ is the target label, z is the

[1] We only take into account characters at the *scene* level, i.e. we do not associate characters with each dialogue statement, leaving this addition to future work.

output of the FFNN, σ is the sigmoid function, and λ_j is the ratio of positive to negative samples (precomputed over the entire training set, since the development set is too small to tune this parameter) for the j^{th} tag label. With this loss function, we account for label imbalance without tuning separate thresholds for each tag on the validation set.

4 Interpreting Scene Embeddings

As the complexity of learning methods used to encode sentences and documents has increased, so has the need to understand the properties of the encoded representations. Probing methods (Linzen et al., 2016; Conneau et al., 2018) gauge the information captured in an embedding by evaluating its performance on downstream classification tasks, either with manually collected annotations (Shi et al., 2016) or self-supervised proxies (Adi et al., 2016). In our case, it is laborious and expensive to collect such annotations at the scene level (requiring domain experts), and the proxy evaluation tasks proposed in literature do not probe the narrative properties we wish to surface.

Instead, we take inspiration from Iyyer et al. (2016) to learn an unsupervised **scene descriptor model** that can be trained without relying on such annotations. Using a dictionary learning technique (Olshausen and Field, 1997), the model learns to represent each scene embedding as a weighted mixture of various topics estimated over the entire corpus. It thus acts as an 'interpretability layer' that can be applied over the scene encoder. This model is similar in spirit to dynamic topic models (Blei and Lafferty, 2006), with the added advantage of producing topics that are both more coherent and more interpretable than those generated by LDA (He et al., 2017; Mitcheltree et al., 2018).

4.1 Scene Descriptor Model

The model has three main components: a **scene encoder** whose outputs we wish to interpret, a set of topics or **descriptors** that are the 'basis elements' used to describe an interpretable scene, and a **predictor** that predicts weights over descriptors for a given scene embedding. The scene encoder uses the text of a given scene s_t to produce a corresponding scene embedding $\mathbf{v}_t$. This encoder can take any form – from an extractor that derives a hand-crafted feature set from the scene text, as in Gorinski and Lapata (2018), to the scene encoder in §3.

To probe the contents of scene embedding $\mathbf{v}_t$, we

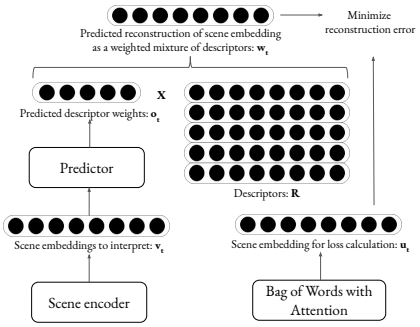

Figure 3: A pictorial representation of the descriptor model.

compute a descriptor-based representation $\mathbf{w}_t \in \mathbb{R}^d$ in terms of a descriptor matrix $\mathbf{R} \in \mathbb{R}^{k \times d}$ that stores k topics or descriptors:

$$\begin{aligned} \mathbf{o}_t &= \text{softmax}(f(\mathbf{v}_t)) \\ \mathbf{w}_t &= \mathbf{R}^T \mathbf{o}_t \end{aligned} \tag{6}$$

where $\mathbf{o}_t \in \mathbb{R}^k$ is the weight (probability) vector over k descriptors and $f(\mathbf{v}_t)$ is a predictor (illustrated by the leftmost pipeline in Fig. 3) which converts $\mathbf{v}_t$ into $\mathbf{o}_t$. Two variants are $f = \text{FFNN}(\mathbf{v}_t)$ and $f = \text{FFNN}([\mathbf{v}_t; \mathbf{o}_{t-1}])$ (concatenation); we use the former in §5.5. Furthermore, we can incorporate additional recurrence into the model by modifying Eq. 6 to add the previous state:

$$\begin{aligned} \mathbf{o}_t =& (1-\alpha) \cdot \text{softmax}(\text{FFNN}([\mathbf{v}_t; \mathbf{o}_{t-1}])) \\ &+ \alpha \cdot \mathbf{o}_{t-1} \end{aligned} \tag{7}$$

Descriptors are initialized either randomly (Glorot and Bengio, 2010) or with the centroids of a k-means clustering of the input word embeddings. For the predictor, f is a two-layer FFNN with ReLU activations and a softmax layer that transforms $\mathbf{v}_t$ (from the scene encoder) into a 100-dimensional intermediate state and then into $\mathbf{o}_t$.

4.2 Reconstruction Task

We wish to minimize the reconstruction error between two scene representations: (1) the descriptor-based embedding $\mathbf{w}_t$ which depends on the scene embedding $\mathbf{v}_t$, and (2) an attention-weighted bag-of-words embedding for s_t. This encourages the computed descriptor weights to be indicative of the scene's actual content (the portions of its text that indicate attributes of interest such as genre, plot, and mood). We use a `BoE+Attn` scene encoder (§3.2) pretrained on the tag classification task (bottom right of Fig. 3), which yields a vector $\mathbf{u}_t \in \mathbb{R}^d$ for scene s_t. The scene descriptor model

is then trained using a hinge loss objective (Weston et al., 2011) to minimize the reconstruction error between $\mathbf{w_t}$ and $\mathbf{u}_t$, with an additional orthogonality constraint on $\mathbf{R}$ to encourage semantically distinct descriptors:

$$L = \Sigma_{j=1}^{n} \max(0, 1 - \mathbf{w}_t^T \mathbf{u}_t + \mathbf{w}_t^T \mathbf{u}_j) + \lambda \|\mathbf{R}\mathbf{R}^T - \mathbf{I}\|_2 \quad (8)$$

where $\mathbf{u}_1 \ldots \mathbf{u}_n$ are n negative samples selected from other scenes in the same screenplay.

We use a `BoE+Attn` scene encoder as a "target" $\mathbf{u}_t$ to force $\mathbf{w}_t$ (and therefore the rows in $\mathbf{R}$) in the same space as the input word embeddings. Thus, a given descriptor can be semantically interpreted by querying its nearest neighbors in the word embedding space. The predicted descriptor weights for a scene s_t are obtained by running a forward pass through the model.

5 Evaluation

We evaluate the proposed script encoder and its variants through two supervised multilabel tag prediction tasks, and a qualitative analysis via the unsupervised extraction of descriptor trajectories.

5.1 Datasets

We base our evaluation on the ScriptBase-J corpus released by Gorinski and Lapata (2018) to directly compare our approach with the multilabel encoder proposed in Gorinski and Lapata (2018) and to provide an open-source evaluation standard.[2] In this corpus, each movie is associated with a set of expert-curated tags that range across 6 tag attributes: mood, plot, genre, attitude, place, and flag; in addition, we also evaluate on an internal dataset of labels assigned to the same movies by in-house domain experts, across 3 tag attributes: genre, plot, and mood. The two taxonomies are distinct. (See Appendix Table 3).

Script Preprocessing

As in Pavel et al. (2015), we leverage the standard screenplay format (Argentini, 1998) to extract structured representations of scripts (formatting cues included capitalization and tab-spacing; see Fig. 1 and Table 1 for an example). Filtering erroneously processed scripts removes 6% of the corpus, resulting in a total of 857 scripts. We hold out 20% (172) scripts for evaluation and use the rest for training. The average number of tokens per script is around 23k; additional statistics are shown in Appendix Table 1.

To keep within GPU memory limits, we split extremely long scenes to retain no more than 60 action and 60 dialogue lines per scene. The vocabulary is composed of words with at least 5 occurrences across the script corpus. The number of scripts per tag value ranges from high (e.g. for some Genre tags) to low (for most Plot and Mood tags) in both datasets (see Appendix Table 4), which along with high tag cardinality for each attribute motivates the use of the reweighted loss in Eq. 5.

5.2 Experimental Setup

All inputs to the hierarchical scene encoder are 100-dimensional GloVe embeddings (Pennington et al., 2014).[3] Our sequential models are bi-GRUs with a single 50-dimensional hidden layer in each direction, resulting in 100-dimensional outputs. The attention parameter $\mathbf{p}$ is 100-dimensional; BoE models naturally output 100-dimensional representations, and character embeddings are 10-dimensional. The script encoder's output is passed through a linear layer with sigmoid activation and binarized by thresholding at 0.5.

One simplification we use is to utilize the same encoder *type* for all encoders described in §3.1. However, particular encoder types might suit different tiers of the architecture: e.g. scene embeddings could be aggregated in a permutation-invariant manner, since narratives are interwoven and scenes may not be truly sequential.

We implement the script encoder on top of AllenNLP (Gardner et al., 2017) and PyTorch (Paszke et al., 2019), and all experiments are conducted on an AWS `p2.8xlarge` machine. We use the Adam optimizer with an initial learning rate of 0.005, clip gradients at a maximum norm of 5, and use no dropout. The model is trained for up to 20 epochs to maximize average precision score, with early stopping if the validation metric does not improve for 5 consecutive epochs.

5.3 Tag Prediction Experiments

ScriptBase-J also comes with loglines, or short, 1-2 sentence human-crafted summaries of the movie's plot and mood (see Appendix Table 2). A model

[2] https://github.com/EdinburghNLP/scriptbase

[3] Using richer contextual word representations will improve performance, but is orthogonal to the purpose of this work.

trained on these summaries can be expected to provide a reasonable baseline for tag prediction, since logline curators are likely to highlight information relevant to this task. The `Loglines` model is a bi-GRU with inputs of size 100 (GloVe embeddings) and hidden units of size 50 in each direction, whose output feeds into a linear classifier.[4]

Model	Genre	Plot	Mood
Loglines	49.9 (0.8)	12.7 (0.9)	17.5 (0.2)
Comparing encoder variations:			
BoE	49.0 (1.1)	8.3 (0.6)	12.9 (0.7)
BoE + Attn	51.9 (2.3)	11.3 (0.4)	16.3 (0.6)
GRU	57.9 (1.9)	13.0 (1.3)	19.1 (1.0)
GRU + Attn	60.5 (2.0)	**15.2** (0.4)	**22.9** (1.4)
Variants on GRU + Attn for action & dialog:			
+ Chars	**62.5** (0.7)	11.7 (0.3)	18.2 (0.3)
- Action	60.5 (2.9)	13.5 (1.4)	20.0 (1.2)
- Dialogue	60.5 (0.6)	13.4 (1.7)	19.1 (1.4)
2-tier	61.3 (2.3)	13.7 (1.7)	20.6 (1.2)
HAN	61.5 (0.6)	14.2 (1.7)	20.7 (1.4)

Table 2: Investigation of the effects of different architectural (BoE +/- Attn, GRU +/- Attn) and structural choices on a tag prediction task, using an internally tagged dataset: F-1 scores with sample standard deviation in parentheses. Across the 3 tag attributes we find that modeling sentential and scene-level structure helps, and attention helps extract representations more salient to the task at hand.

Table 2 contains results for the tag prediction task on our internally-tagged dataset. First, a set of models trained using action and dialogue inputs are used to evaluate the architectural choices in §3.1. We find that modeling recurrence at sentential and scene levels and selecting relevant words/scenes with attention are prominent factors in the robust improvement over the `Loglines` baseline (see the first five rows in Table 2).

Next, we assess the effect that various structural elements of a screenplay have on classification performance. Notably, the difficulty of the prediction task is directly related to the number of labels per tag attribute: higher-cardinality tag attributes with correlated tag values (like plot and mood) are far more difficult to predict than lower-cardinality tags with more discriminable values (like genre). We find that adding character information to the best-performing GRU + Attn model (`+Char`) improves prediction of genre, while using both dialogue and action statements improves performance on plot and mood when compared to using only one or the other. We also evaluate (1) a `2-tier` variant of the `GRU+Attn` model without action/dialogue-statement encoders (i.e., all action statements are concatenated into a single sequence of words and passed into the action-scene encoder, and similarly with dialogue) and (2) a variant similar to Yang et al. (2016) (`HAN`) that does not distinguish between action and dialogue (i.e., all statements in a scene are encoded using a single statement encoder and statement embeddings are passed to a scene encoder, the output of which is passed into the script encoder). Both models perform slightly better than `GRU+Attn` on genre, but worse on plot and mood, indicating that incorporating hierarchy and distinguishing between dialogue and action statements helps on the more difficult prediction tasks.

Tag	G&L	HSE
Attitude	72.6	70.1
Flag	52.5	52.6
Genre	55.1	42.5
Mood	45.5	51.2
Place	57.7	29.1
Plot	34.6	34.5

Table 3: F-1 scores on ScriptBase-J provided tag set, comparing Gorinski and Lapata (2018)'s approach to ours.

For the results in Table 3, we compared the `GRU+Attn` configuration in Table 2 (`HSE`) with an implementation of Gorinski and Lapata (2018) (`G&L`) that was run on the previous train-test split. `G&L` contains a number of handcrafted lexical, graph-based, and interactive features that were designed for optimal performance on screenplay analysis. In contrast, `HSE` directly encodes standard screenplay structure into a neural network architecture, and is an alternative, arguably more lightweight way of building a domain-specific textual representation. Our results are comparable, with the exception of 'place', which can often be identified deterministically from scene headings.

5.4 Similarity-based F-1

Results in Tables 2 and 3 check for an exact match between predicted and true tag values to report standard multi-label F-1 scores (one-vs-rest classification evaluation, micro-averaged over tag attributes). However, the characteristics of our tag taxonomies suggest that this measure may not be ideal, since human-crafted tag sets include dozens of highly correlated, overlapping values, and the dataset includes instances of missing tags. A standard scoring procedure may underestimate model

[4] We tried both with and without attention and found the variant without attention to give slightly better results.

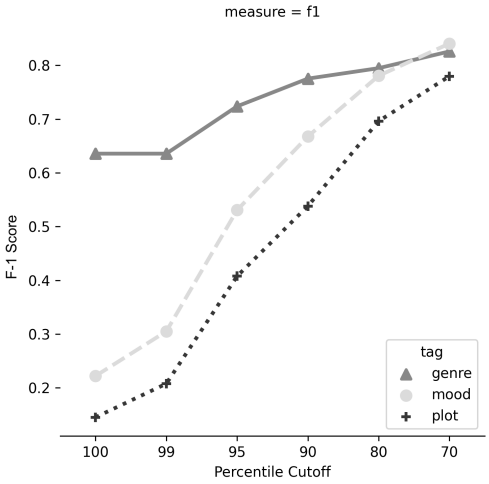

Figure 4: F1 score of various tag attributes as a function of the similarity threshold percentile.

performance when, e.g., a prediction of 'Crime' is equally penalized for a target labels of 'Heist' and 'Romance' (see Appendix Table 5).

We use a similarity-based scoring procedure (see Maynard et al. (2006) for related approaches) to assess the impact of such effects. In particular, we calculate cosine similarities between tag embeddings trained on a similar task (see Appendix for details) and evaluate a prediction based the percentile of its similarity to the actual label. Such a measure takes into account the latent relationships among tags via *similarity thresholding*, wherein a prediction is counted as correct if it is sufficiently similar to the target. The percentile cutoff can be varied to estimate model performance as a function of the threshold percentile.

In Fig. 4 we re-evaluate the `GRU + Attn` model outputs (row 5 in Table 2) with this evaluation metric to examine how our results might vary if we adopted a similarity-based scoring procedure. When the similarity percentile cutoff equals 100, the result is identical to the standard F-1 score. Even decreasing the cutoff to the 90$^{\text{th}}$ percentile shows striking improvements for high-cardinality attributes (180% for `mood` and 250% for `plot`). Notably, using a similarity-based scoring procedure for complex tag taxonomies may yield results that more accurately reflect human perception of the model's performance (Maynard et al., 2006).

5.5 Qualitative Scene-level Analysis

To extract narrative trajectories with the scene descriptor model, we analyze the scene encoder from the `GRU+Attn` model, which performs best on the Plot and Mood tag attributes and does reasonably well on Genre. Similarly to Iyyer et al. (2016), we limit the input vocabulary for the `BoE+Attn` encoders that yield target vectors $\mathbf{u}_t$ to words occurring in at least 50 movies (7.3% of the training set), while also filtering the 500 most frequent words in the corpus. We set the number of descriptors k to 25 to allow for a wide range of topics while keeping manual examination feasible.

Further modeling choices are evaluated using the semantic coherence metric (Mimno et al., 2011), which assesses the quality of word clusters induced by topic modeling algorithms. These choices include: the presence of recurrence in the predictor (i.e. toggling between Eqns. 6 and 7, with $\alpha = 0.5$) and the value of hyperparameter λ. While the k-means initialized descriptors score slightly higher on semantic coherence, they remain close to the initial centroids and do not reflect the corpus as well as the randomly initialized version, which is the initialization we eventually used. We also find that incorporating recurrence and $\lambda = 10$ (tuned using simple grid search) result in the highest coherence.

The outputs of the scene descriptor model are shown in Table 4 and Figure 5. Table 4 presents five example descriptors, each identified by the representative words closest to them in the word embedding space (topic names are manually annotated). Figure 5 presents the narrative trajectories of a subset of descriptors for three screenplays: Pretty Woman, Pulp Fiction, and Pearl Harbor, using a *streamgraph* (Byron and Wattenberg, 2008). The descriptor weight $\mathbf{o}_t$ (Eq. 6) as a function of scene number/order is rescaled and smoothed, with the width of a color band indicating the weight value. A critical event for each screenplay is indicated by a letter on each trajectory. A qualitative analysis of such events indicates general alignment between scripts and their topic trajectories, and the potential applicability of this method to identifying significant moments in long-form documents.

Topic	**Words**
Violence	fires blazes explosions grenade blasts
Residential	loft terrace courtyard foyer apartments
Military	leadership army victorious commanding elected
Vehicles	suv automobile wagon sedan cars
Geography	sand slope winds sloping cliffs

Table 4: Examples of retrieved descriptors. Trajectories for "Violence", "Military", and "Residential" are shown in Fig. 5.

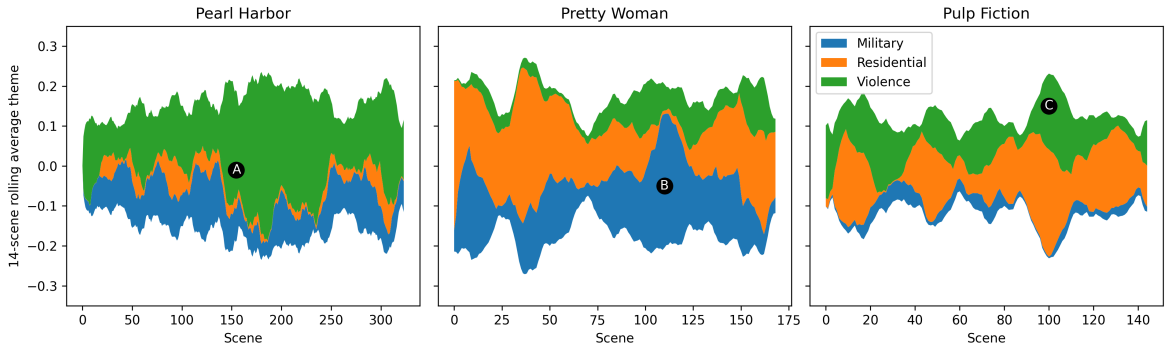

Figure 5: Descriptor Trajectories for *Pearl Harbor*, *Pretty Woman*, and *Pulp Fiction*. The y-axis is a smoothed and rescaled descriptor weight, i.e. $\mathbf{o}_t$ in Eq. 6. Events: (A) Attack on Pearl Harbor begins (B) Rising tension at the equestrian club and (C) Confrontation at the pawn shop. Word clusters corresponding to each descriptor are in Table 4.

6 Related Work

Computational narrative analysis of large texts has been explored in a range of contexts (Mani, 2012) over the past few decades (Lehnert, 1981). Recent work has analyzed narrative from plot (Chambers and Jurafsky, 2008; Goyal et al., 2010) and character (Elsner, 2012; Bamman et al., 2014) perspectives. While movie narratives have received attention (Bamman et al., 2013; Chaturvedi et al., 2018; Kar et al., 2018), the computational analysis of entire screenplays is not as common.

Notably, Gorinski and Lapata (2015) introduced a summarization method for scripts, extracting graph-based features that summarize the key scene sequences. Gorinski and Lapata (2018) built on top of this work, crafting additional features for a specially-designed multilabel encoder, while also emphasizing the difficulty of the tag prediction task. Our work suggests an orthogonal approach using automatically learned scene representations instead of feature-engineered inputs. We also consider the possibility that at least some of the task difficulty owes not to the length or richness of the text, but rather to the complexity of the tag taxonomy. The pattern of results we obtain from a similarity-based scoring measure offers a brighter picture of model performance, and suggests that the standard multilabel F1 measure may not be appropriate for such complex tag sets (Maynard et al., 2006).

Nevertheless, dealing with long-form text remains a significant challenge. One possible solution is to infer richer representations of latent structure using a structured attention mechanism (Liu and Lapata, 2018), which might highlight key dependencies between scenes in a script. Another method could be to define auxiliary tasks as in Jiang and Bansal (2018) to encourage better selection. Lastly, sparse versions of the softmax function (Martins and Astudillo, 2016) could be used to address the sparse distribution of salient information across a screenplay.

7 Conclusion

In this work, we propose and evaluate various neural network architectures for learning fixed-dimensional representations of full-length film scripts. We hypothesize that a network design mimicking the documents' internal structure will boost performance. Experiments on two tag prediction tasks support this hypothesis, confirming the benefits of using hierarchical attention-based models and of incorporating distinctions between various scene components directly into the model. In order to explore the information contained within scene-level embeddings, we present an unsupervised technique for bootstrapping scene "descriptors" and visualizing their trajectories over the course of the screenplay. For future work, we plan to investigate richer ways of representing character identities, which could allow character embeddings to be compared across movies and linked to character archetypes. A persona-based characterization of the screenplay would provide a complementary view to the current plot-based analysis.

Scripts and screenplays are an underutilized and underanalyzed data source in modern NLP - indeed, most work on narratology in NLP concentrates on short stories and book/movie summaries. This paper shows that capitalizing on their rich internal structure largely obviates the need for feature-engineering, or other more complicated architectures, a lesson that may prove instructive in other areas of discourse processing. Our hope is that these results encourage more people to work on this fascinating domain.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and feedback, and Ashish Rastogi for his support and guidance.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2016. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *Proceedings of ICLR*.

Paul Argentini. 1998. *Elements of Style for Screenwriters*. Lone Eagle Publishing.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

David Bamman, Brendan O'Connor, and Noah A. Smith. 2013. Learning latent personas of film characters. In *Proceedings of ACL*.

David Bamman, Ted Underwood, and Noah A. Smith. 2014. A bayesian mixed effects model of literary character. In *Proceedings of ACL*.

Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, and Marcos Zampieri. 2019. Findings of the 2019 conference on machine translation (wmt19). In *Proceedings of WMT*.

Chandra Bhagavatula, Sergey Feldman, Russell Power, and Waleed Ammar. 2018. Content-based citation recommendation. In *Proceedings NAACL*.

David M. Blei and John D. Lafferty. 2006. Dynamic topic models. In *Proceedings of ICML*.

L Byron and M. Wattenberg. 2008. Stacked graphs – geometry aesthetics. *IEEE Transactions on Visualization and Computer Graphics*.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL*.

Snigdha Chaturvedi, Shashank Srivastava, and Dan Roth. 2018. Where have I heard this story before? identifying narrative similarity in movie remakes. In *Proceedings of NAACL*.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. *Proceedings of ACL*.

Micha Elsner. 2012. Character-based kernels for novelistic plot structure. In *Proceedings of EACL*.

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

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of AIStats*.

Philip John Gorinski and Mirella Lapata. 2015. Movie script summarization as graph-based scene extraction. In *Proceedings of NAACL*.

Philip John Gorinski and Mirella Lapata. 2018. What's this movie about? a joint neural network architecture for movie content analysis. In *Proceedings of NAACL*.

Amit Goyal, Ellen Riloff, and Hal Daumé, III. 2010. Automatically producing plot unit representations for narrative text. In *Proceedings of EMNLP*.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2017. An unsupervised neural attention model for aspect extraction. In *Proceedings of ACL*.

James Henderson. 2020. The unstoppable rise of computational linguistics in deep learning. In *Proceedings of ACL*.

Mohit Iyyer, Anupam Guha, Snigdha Chaturvedi, Jordan Boyd-Graber, and Hal Daumé III. 2016. Feuding families and former Friends: Unsupervised learning for dynamic fictional relationships. In *Proceedings of NAACL*.

Yangfeng Ji and Noah A. Smith. 2017. Neural discourse structure for text categorization. In *Proceedings of ACL*.

Yichen Jiang and Mohit Bansal. 2018. Closed-book training to improve summarization encoder memory. In *Proceedings of EMNLP*.

Sudipta Kar, Suraj Maharjan, and Thamar Solorio. 2018. Folksonomication: Predicting tags for movies from plot synopses using emotion flow encoded neural network. In *Proceddings of COLING*.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The NarrativeQA reading comprehension challenge. *Transactions of the ACL*.

Wendy G. Lehnert. 1981. Plot units and narrative summarization. *Cognitive Science*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the ACL*.

Yang Liu and Mirella Lapata. 2018. Learning structured text representations. *Transactions of the Association for Computational Linguistics*.

Inderjeet Mani. 2012. *Synthesis Lectures on Human Language Technologies: Computational Modeling of Narrative*. Morgan Claypool.

Andre Martins and Ramon Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *Proceedings of ICML*.

Diana Maynard, Wim Peters, and Yaoyong Li. 2006. Metrics for evaluation of ontology-based information extraction. In *CEUR Workshop Proceedings*.

David Mimno, Hanna M Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of EMNLP*.

Christopher Mitcheltree, Skyler Wharton, and Avneesh Saluja. 2018. Using aspect extraction approaches to generate review summaries and user profiles. In *Proceedings of NAACL*.

Bruno A Olshausen and David J Field. 1997. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Proceedings of NeurIPS*.

Amy Pavel, Dan B. Goldman, Björn Hartmann, and Maneesh Agrawala. 2015. Sceneskim: Searching and browsing movies using synchronized captions, scripts and plot summaries. In *Proceedings of UIST*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*.

Xiang Ren, Jialu Liu, Xiao Yu, Urvashi Khandelwal, Quanquan Gu, Lidan Wang, and Jiawei Han. 2014. Cluscite: Effective citation recommendation by information network-based clustering. In *Proceedings of KDD*.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *Proceedings of EMNLP*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of EMNLP*.

Sainbayar Sukhbaatar, arthur szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Proceedings of NeurIPS*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of ICLR*.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *Proceedings of IJCAI*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of NAACL*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of NAACL*.

A Appendix

A.1 Additional Dataset Statistics

In this section, we present additional statistics on the evaluation sets used in this work.

Min	10th %	90th %	Max
4025	16,240	29,376	52,059

Table 5: Statistics on the number of tokens per script in the Scriptbase-J corpus. We use the same script corpus with two different tag sets – the Jinni tags provided with ScriptBase and a tag set designed by internal annotators.

Tag	Value
Genre	Crime, Independent
Mood	Clever, Witty, Stylized
Attitude	Semi Serious, Realistic
Plot	Tough Heroes, Violence Spree, On the Run
Place	California, Los Angeles, Urban
Flag	Drugs/Alcohol, Profanity, Violent Content
Logline	"The lives of two mob hit men, a boxer, a gangster's wife, and a pair of diner bandits intertwine in four tales of violence and redemption."

Table 6: Examples of Scriptbase-J tag attributes, tag values, and a logline, for the film "Pulp Fiction".

Tag	Internal	Scriptbase-J
Genre	9	31
Mood	65	18
Attitude	-	8
Plot	82	101
Place	-	24
Flag	-	6

Table 7: The number of distinct tag values for each tag attribute across the two datasets. Cardinalities for Scriptbase-J tag attributes are identical to Gorinski and Lapata (2018) except for the removal of one mood tag value when filtering for erroneously preprocessed scripts.

Tag	Avg. #tags/script	Min #scripts/tag	Max #scripts/tag
Genre	1.74	17	347
Mood	3.29	15	200
Plot	2.50	15	73

Table 8: Statistics for the three tag attributes applied in our internally-tagged dataset: average number of tags per script, and the minimum/maximum number of movies associated with any single value.

A.2 Tag Similarity Scoring

To estimate tag-tag similarity percentiles, we calculate the distance between tag embeddings learned via an auxiliary model trained on a related supervised learning task. In our case, the related task is to predict the audience segment of a movie, given a tag set. The general approach is easily replicable via any model that projects tags into a well-defined similarity space (e.g., knowledge-graph embeddings (**?**) or tag-based autoencoders).

Tag	Target	Similar	Unrelated
Genre	Period	Historical	Fantasy
Mood	Witty	Humorous	Bleak
Plot	Hitman	Deadly	Love/Romance

Table 9: Examples of closely related and unrelated tag values in the Scriptbase-J tag set.

Given a tag embedding space, the similarity percentile of a pair of tag values is estimated as follows. For a given tag attribute, the pairwise cosine distance between tag embeddings is computed for all tag-tag value pairs. For a given pair, its similarity percentile is then calculated with reference to the overall distribution for that attribute.

Similarity thresholding simplifies the tag prediction task by significantly reducing the *perplexity* of the tag set, while only marginally reducing its *cardinality*. Cardinality can be estimated via permutations. If n is the cardinality of the tag set, the number of permutations p of different tag pairs ($k = 2$) is:

$$p(n, k) = \frac{n!}{(n-k)!} \tag{9}$$

which simplifies to $n^2 - n - p = 0$.

Likewise, the entropy of a list of n distinct tag values of varying probabilities is given by:

$$H(X) = H(\text{tag}_1, ..., \text{tag}_n) = -\sum_{i=1}^{n} \text{tag}_i \log_2 \text{tag}_i \tag{10}$$

The perplexity over tags is then simply $2^{H(X)}$.

Tag	Perplexity	Cardinality
Genre	42%	16%
Mood	77%	16%
Plot	79%	16%

Table 10: The percent decrease in perplexity and cardinality, respectively, as the similarity threshold decreases from 100th percentile similarity (baseline) to 70th percentile.

As the similarity threshold decreases, the number of tags treated as equivalent correspondingly increases. Mapping these "equivalents" to a shared label in our list of tag values allows us to calculate updated values for tag (1) perplexity and (2)

cardinality. As illustrated by Table 10, rather than leading to large reductions in the overall cardinality of the tag set, similarity thresholding mainly serves to decrease perplexity by eliminating redundant/highly similar alternatives. Thus, thresholding at once significantly decreases the complexity of the prediction task, while yielding a potentially more representative picture of model performance.