Feuding Families and Former Friends: Unsupervised Learning for Dynamic Fictional Relationships

Mohit Iyyer,[1] **Anupam Guha,**[1] **Snigdha Chaturvedi,**[1] **Jordan Boyd-Graber,**[2] **Hal Daumé III**[1]
[1]University of Maryland, Department of Computer Science and UMIACS
[2]University of Colorado, Department of Computer Science
{miyyer,aguha,snigdac,hal}@umiacs.umd.edu,
Jordan.Boyd.Graber@colorado.edu

Abstract

Understanding how a fictional relationship between two characters changes over time (e.g., from best friends to sworn enemies) is a key challenge in digital humanities scholarship. We present a novel unsupervised neural network for this task that incorporates dictionary learning to generate interpretable, accurate relationship trajectories. While previous work on characterizing literary relationships relies on plot summaries annotated with predefined labels, our model jointly learns a set of global relationship descriptors as well as a trajectory over these descriptors for each relationship in a dataset of raw text from novels. We find that our model learns descriptors of events (e.g., marriage or murder) as well as interpersonal states (love, sadness). Our model outperforms topic model baselines on two crowdsourced tasks, and we also find interesting correlations to annotations in an existing dataset.

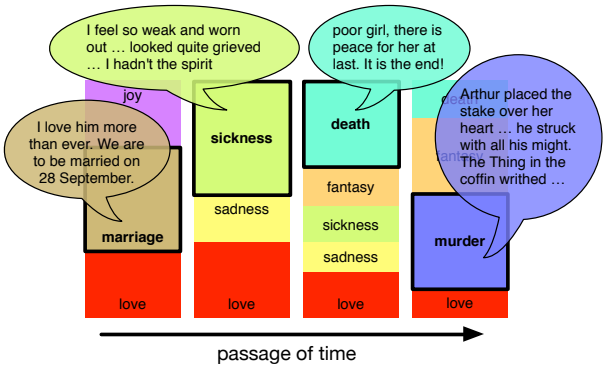


Figure 1: An example trajectory depicting the dynamic relationship between Lucy and Arthur in Bram Stoker's *Dracula*, which starts with love and ends with Arthur killing the vampiric Lucy. Each column describes the relationship state at a particular time by weights over a set of descriptors (larger weights shown as bigger boxes). Our goal is to learn—without supervision—both the descriptors and the trajectories from raw fictional texts.

1 Describing Character Relationships

When two characters in a book break bread, is their meal just a result of biological needs or does it mean more? Cognard-Black et al. (2014) argue that this simple interaction reflects the diversity and background of the characters, while Foster (2009) suggests that the tone of a meal can portend either good or ill for the rest of the book. To support such theories, scholars use their literary expertise to draw connections between disparate books: Gabriel Conroy's dissonance from his family at a sumptuous feast in Joyce's *The Dead*, the frustration of Tyler's mother in *Dinner at the Homesick Restaurant*, and the grudging respect for a blind man eating meatloaf in Carver's *Cathedral*.

However, these insights do not come cheap. It takes years of careful reading and internalization to make connections across books, which means that relationship symmetries and archetypes are likely to remain hidden in the millions of books published every year unless literary scholars are actively searching for them.

Natural language processing techniques have been increasingly used to assist in these literary investigations by discovering patterns in texts (Jockers, 2013). In Section 6 we review existing techniques that classify or cluster relationships between characters in books using a fixed set of labels (e.g., friend or en-

Proceedings of NAACL-HLT 2016, pages 1534–1544,
San Diego, California, June 12-17, 2016.

emy). However, such approaches ignore interactions between characters that lie outside of the established lexicon and cannot account for the dynamic nature of relationships that evolve through the course of a book, such as the vampiric downfall of Lucy and Arthur's engagement in *Dracula* (Figure 1) or Winston Smith's rat-induced betrayal of Julia in *1984*.

To address these issues, we propose the task of unsupervised relationship modeling, in which a model jointly learns a set of *relationship descriptors* as well as *relationship trajectories* for pairs of literary characters. Instead of assigning a single descriptor to a particular relationship, the trajectories learned by the model are sequences of descriptors as in Figure 1.

The Bayesian *hidden topic Markov model* (**HTMM**) of Gruber et al. (2007) emerges as a natural choice for our task because it is capable of computing relationship descriptors (in the form of topics) and has an additional temporal component. However, our experiments show that the descriptors learned by the **HTMM** are not coherent and focus more on events or environments (e.g., meals, outdoors) than interpersonal states like happiness and sadness.

Motivated by recent advances in deep learning, we propose the *relationship modeling network* (**RMN**), which is a novel variant of a deep recurrent autoencoder that incorporates dictionary learning to learn relationship descriptors. We show that the **RMN** achieves better descriptor coherence and trajectory accuracy than the **HTMM** and other topic model baselines in two crowdsourced evaluations described in Section 4. In Section 5 we show qualitative results and make connections to existing literary scholarship.

2 A Dataset of Character Interactions

Our dataset consists of 1,383 fictional works pulled from Project Gutenberg and other Internet sources. Project Gutenberg has a limited selection (outside of science fiction) of mostly classic literature, so we add more contemporary novels from various genres such as mystery, romance, and fantasy to our dataset.

To identify character mentions, we run the BookNLP pipeline of Bamman et al. (2014), which includes character name clustering, quoted speaker identification, and coreference resolution.[1] For every detected character mention, we define a span as beginning 100 tokens before the mention and ending 100 tokens after the mention. We do not use sentence or paragraph boundaries because they vary considerably depending on the author (e.g., William Faulkner routinely wrote single sentences longer than many of Hemingway's paragraphs). All spans in our dataset contain mentions to exactly two characters. This is a rather strict requirement that forces a reduction in data size, but spans in which more than two characters are mentioned are generally noisier.

Once we have identified usable spans in the dataset, we apply a second filtering step that removes relationships containing fewer than five spans. Without this filter, our dataset is dominated by fleeting interactions between minor characters; this is undesirable since our focus is on longer, mutable relationships. Finally, we filter our vocabulary by removing the 500 most frequently occurring words, as well as all words that occur in fewer than 100 books. The latter step helps correct for variation in time period and genre (e.g., "thou" and "thy" found in older works like the *Canterbury Tales*). Our final dataset contains 20,013 relationships and 380,408 spans, while our vocabulary contains 16,223 words.[2]

3 Relationship Modeling Networks

This section mathematically describes how we apply the **RMN** to relationship modeling on our dataset. Our model is similar in spirit to topic models: for an input dataset, the output of the **RMN** is a set of relationship descriptors (topics) and—for each relationship in the dataset—a trajectory, or a sequence of probability distributions over these descriptors (document-topic assignments). However, the **RMN** uses recent advances in deep learning to achieve better control over descriptor coherence and trajectory smoothness (Section 4).

3.1 Formalizing the Problem

Assume we have two characters c_1 and c_2 in book b. We define S_{c_1,c_2} as a sequence of token spans where each span $s_t \in S_{c_1,c_2}$ is itself a set of tokens

[1] While this pipeline works reasonably well, it is unreliable for first-person narratives; we leave the necessary improvements to character name clustering, which are further expanded upon in Vala et al. (2015), for future work.

[2] Code and span data available at `http://github.com/miyyer/rmn`.

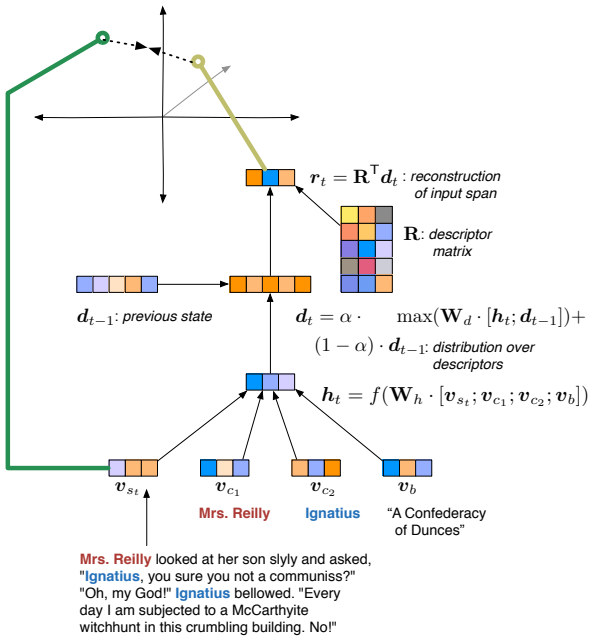


Figure 2: An example of the RMN's computations at a single time step. The model approximates the vector average of an input span ($\boldsymbol{v}_{s_t}$) as a linear combination of descriptors from $\mathbf{R}$. The descriptor weights $\boldsymbol{d}_t$ define the relationship state at each time step and—when viewed as a sequence—form a relationship trajectory.

$\{w_1, w_2, \ldots, w_l\}$ of fixed size l that contains mentions (either directly or by coreference) to both c_1 and c_2. In other words, S_{c_1,c_2} includes the text of every scene, chronologically ordered, in which c_1 and c_2 are present together.

3.2 Model Description

As in other neural network models for natural language processing, we begin by associating each word type w in our vocabulary with a real-valued embedding $\boldsymbol{v}_w \in \mathbb{R}^d$. These embeddings are rows of a $V \times d$ matrix $\mathbf{L}$, where V is the vocabulary size. Similarly, characters and books have their own embeddings in rows of matrices $\mathbf{C}$ and $\mathbf{B}$. We want $\mathbf{B}$ to capture global context information (e.g., "Moby Dick" takes place at sea) and $\mathbf{C}$ to capture immutable aspects of characters not related to their relationships (e.g., Javert is a police officer). Finally, the RMN learns embeddings for relationship descriptors, which requires a second matrix $\mathbf{R}$ of size $K \times d$ where K is the number of descriptors, analogous to the number of topics in topic models.

Each input to the RMN is a tuple that contains identifiers for a book and two characters, as well as the spans corresponding to their relationship: $(b, c_1, c_2, S_{c_1,c_2})$. Given one such input, our objective is to reconstruct S_{c_1,c_2} using a linear combination of relationship descriptors from $\mathbf{R}$ as shown in Figure 2; we now describe this process formally.

3.2.1 Modeling Spans with Vector Averages

We compute a vector representation for each span s_t in S_{c_1,c_2} by averaging the embeddings of the words in that span,

$$\boldsymbol{v}_{s_t} = \frac{1}{l} \sum_{w \in s_t} \boldsymbol{v}_w. \quad (1)$$

Then, we concatenate $\boldsymbol{v}_{s_t}$ with the character embeddings $\boldsymbol{v}_{c_1}$ and $\boldsymbol{v}_{c_2}$ as well as the book embedding $\boldsymbol{v}_b$ and feed the resulting vector into a standard feed-forward layer to obtain a hidden state $\boldsymbol{h}_t$,

$$\boldsymbol{h}_t = f(\mathbf{W}_h \cdot [\boldsymbol{v}_{s_t}; \boldsymbol{v}_{c_1}; \boldsymbol{v}_{c_2}; \boldsymbol{v}_b]). \quad (2)$$

In all experiments, the transformation matrix W_h is $d \times 4d$, and we set f to the ReLu function, $\text{ReLu}(x) = \max(0, x)$.

3.2.2 Approximating Spans with Relationship Descriptors

Now that we can obtain representations of spans, we move on to learning descriptors using a variant of dictionary learning (Olshausen and Field, 1997; Elad and Aharon, 2006), where our descriptor matrix $\mathbf{R}$ is the dictionary and we are trying to approximate input spans as a linear combination of items from this dictionary.

Suppose we compute a hidden state for every span s_t in S_{c_1,c_2} (Equation 2). Now, given an $\boldsymbol{h}_t$, we compute a weight vector $\boldsymbol{d}_t$ over K relationship descriptors with some composition function g, which is fully specified in the next section. Conceptually, each $\boldsymbol{d}_t$ is a *relationship state*, and a *relationship trajectory* is a sequence of chronologically-ordered relationship states as shown in Figure 1. After computing $\boldsymbol{d}_t$, we use it to compute a reconstruction vector $\boldsymbol{r}_t$ by taking a weighted average over relationship descriptors,

$$\boldsymbol{r}_t = \mathbf{R}^\top \boldsymbol{d}_t. \quad (3)$$

Our goal is to make $\boldsymbol{r}_t$ similar to $\boldsymbol{v}_{s_t}$. We use a contrastive max-margin objective function similar to

previous work (Weston et al., 2011; Socher et al., 2014). We randomly sample spans from our dataset and compute the vector average $\boldsymbol{v}_{s_n}$ for each sampled span as in Equation 1. This subset of span vectors is N. The unregularized objective J is a hinge loss that minimizes the inner product between $\boldsymbol{r}_t$ and the negative samples while simultaneously maximizing the inner product between $\boldsymbol{r}_t$ and $\boldsymbol{v}_{s_t}$,

$$J(\theta) = \sum_{t=0}^{|S_{c_1,c_2}|} \sum_{n \in N} \max(0, 1 - \boldsymbol{r}_t \boldsymbol{v}_{s_t} + \boldsymbol{r}_t \boldsymbol{v}_{s_n}), \quad (4)$$

where θ represents the model parameters.

3.2.3 Computing Weights over Descriptors

What function should we choose for our composition function g to represent a relationship state at a given time step? On the face of it, this seems trivial; we can project h_t to K dimensions and then apply a softmax or some other nonlinearity that yields non-negative weights.[3] However, this method ignores the relationship states at previous time steps. To model the temporal aspect of relationships, we can add a recurrent connection,

$$\boldsymbol{d}_t = \text{softmax}(\mathbf{W}_d \cdot [\boldsymbol{h}_t; \boldsymbol{d}_{t-1}]) \quad (5)$$

where $\mathbf{W}_d$ is of size $K \times (d + K)$ and $\text{softmax}(\boldsymbol{q}) = {\exp \boldsymbol{q}}/{\sum_{j=1}^{k} \exp \boldsymbol{q}_j}$.

Our hope is that this recurrent connection will carry some of the previous relationship state over to the current time step. It should be unlikely for two characters in love at time t to fall out of love at time $t + 1$ even if s_{t+1} does not include any love-related words. However, because the objective function in Equation 4 maximizes similarity with the current time step's input, the model is not forced to learn a smooth interpolation between the previous state and the current one. A natural remedy is to have the model predict the *next* time step's input instead, but this proves hard to optimize.

We instead *force* the model to use the previous relationship state by modifying Equation 5 to include a linear interpolation between $\boldsymbol{d}_t$ and $\boldsymbol{d}_{t-1}$,

$$\begin{aligned} \boldsymbol{d}_t = \alpha \cdot \text{softmax}(\mathbf{W}_d \cdot [\boldsymbol{h}_t; \boldsymbol{d}_{t-1}]) + \\ (1 - \alpha) \cdot \boldsymbol{d}_{t-1}. \end{aligned} \quad (6)$$

[3] We experiment with a variety of nonlinearities but find that the softmax yields the most interpretable results due to its predisposition to select a single descriptor.

Here, α is a scalar between 0 and 1. We experiment with setting α to a fixed value of 0.5 as well as allowing the model to learn α as in

$$\alpha = \sigma(\boldsymbol{v}_\alpha^\mathsf{T} \cdot [\boldsymbol{h}_t; \boldsymbol{d}_{t-1}; \boldsymbol{v}_{s_t}]), \quad (7)$$

where σ is the sigmoid function and $\boldsymbol{v}_\alpha$ is a vector of dimensionality $2d + K$. Fixing $\alpha = 0.5$ initially and then tuning it after other parameters have converged improves training stability; for the specific hyperparameters we use see Section 4.[4]

3.2.4 Interpreting Descriptors and Enforcing Uniqueness

Recall that each descriptor is a d-dimensional row of $\mathbf{R}$. Because our objective function J forces these descriptors to be in the same vector space as that of the word embeddings $\mathbf{L}$, we can interpret them by looking at nearest neighbors in $\mathbf{L}$ using cosine distance as the similarity metric.

To discourage learning descriptors that are too similar to each other, we add another penalty term X to our objective function,

$$X(\theta) = \left\| \mathbf{R}\mathbf{R}^\mathsf{T} - \mathbf{I} \right\|, \quad (8)$$

where $\mathbf{I}$ is the identity matrix. This term comes from the component orthogonality constraint in independent component analysis (Hyvärinen and Oja, 2000).

We add J and X together to obtain our final training objective L,

$$L(\theta) = J(\theta) + \lambda X(\theta), \quad (9)$$

where λ is a hyperparameter that controls the magnitude of the uniqueness penalty.

4 Evaluating Descriptors and Trajectories

Because no previous work explores the interpretability of unsupervised relationship modeling over time, evaluating the **RMN** is tricky. Further compounding the problem is the subjective nature of the task; for example, is a trajectory that ignores a key event better than one that hallucinates episodes absent from source text?

[4] This strategy is reminiscent of alternative minimization strategies for dictionary learning (Agarwal et al., 2014), where the dictionary and weights are learned separately by keeping the other fixed.

With these issues in mind, we conduct three evaluations to show that our output is reasonable. First, we conduct a crowdsourced interpretability experiment that shows **RMN**s produce significantly more coherent descriptors than three topic model baselines. A second crowdsourced task indicates that our model produces trajectories that match plot summaries more accurately than topic models. Finally, we qualitatively compare the **RMN**'s output to existing static annotations of literary relationships and find both expected and surprising results.

4.1 Topic Model Baselines

Before moving onto the evaluations, we briefly describe three baseline models, all of which are Bayesian generative models. Latent Dirichlet allocation (Blei et al., 2003, **LDA**) learns a single document-topic distribution per document; we can apply **LDA** to our dataset by concatenating all spans from a relationship into a single document. Similarly, **NUBBI** (Chang et al., 2009a) learns separate sets of topics for relationships and individual characters.[5]

LDA and **NUBBI** are incapable of taking into account the chronological ordering of the spans because they view all relationships tokens as exchangeable. While we can compare the *descriptors* learned by these models to those of the **RMN**, we cannot evaluate their *trajectories*. We turn instead to the hidden topic Markov model (Gruber et al., 2007, **HTMM**), which foregoes the bag-of-words assumption of **LDA** and **NUBBI** in favor of modeling topic segments within a document as a Markov chain. This model outputs a smooth sequence of topic assignments over a document, so we can compare the *trajectories* it learns on our dataset to those of the **RMN**.

4.2 Experimental Settings

In our descriptor interpretability experiments, we vary the number of descriptors (topics) for all models ($K = 10, 30, 50$). We train **LDA** and **NUBBI** for 100 iterations with a collapsed Gibbs sampler, and the **HTMM** uses the default setting of 100 EM iterations.

For the **RMN**, we initialize the word embedding matrix $\mathbf{L}$ with 300-dimensional GloVe embeddings trained on the Common Crawl (Pennington et al., 2014). The character and book embeddings ($\mathbf{C}$ and $\mathbf{B}$) are initialized randomly. We fix α to 0.5 for the first 15 epochs of training; after the descriptor matrix $\mathbf{R}$ has converged, we fix $\mathbf{R}$ and tune α using Equation 6 for 15 more epochs.[6] Since the topic model baselines do not have access to character and book metadata, for fair comparison we also train a "generic" version of the **RMN** (**GRMN**) where the metadata embeddings are removed from Equation 2. The uniqueness penalty λ is set to 10^{-4}.

All of the **RMN** model parameters except $\mathbf{L}$ are optimized using Adam (Kingma and Ba, 2014) with a learning rate of 0.001 for 30 epochs; the word embeddings are not fine-tuned during training.[7] We also apply word dropout (Iyyer et al., 2015) to the input spans, removing words from the vector average computation in Equation 1 with probability 0.5.

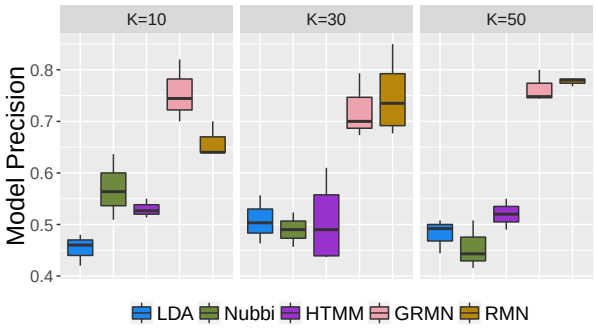


Figure 3: Model precision results from our word intrusion task. The **RMN** learns more interpretable descriptors than three topic model baselines.

4.3 Do Descriptors Make Sense?

The goal of our first experiment is to compare the descriptors $\mathbf{R}$ learned by the **RMN** to the topics learned by the topic model baselines. We conduct a word intrusion experiment (Chang et al., 2009b): workers identify an "intruder" word from a set of words that—other than the intruder—come from the same topic. For the topic models, the five most probable words are joined by a highly-probable word from a different topic as the intruder. We use the same procedure for the **RMN** and **GRMN**, except that cosine similarity to

[5] **NUBBI** requires additional spans that mention only a single character to differentiate character topics from relationship topics. None of the other models receives these extra data.

[6] Preliminary experiments show that learning α and $\mathbf{R}$ simultaneously results in less interpretable descriptors.

[7] Tuning $\mathbf{L}$ ruins descriptor interpretability; pretrained embeddings are likely already a good solution for our problem.

RMN			HTMM		
Label	**MP**	**Nearest Neighbors**	**Label**	**MP**	**Most Probable Words**
sadness	1.0	regretful rueful pity pained despondent	violence	1.0	sword shot blood shouted swung
love	1.0	love delightful happiness enjoyed	boats	1.0	ship boat captain deck crew
murder	1.0	autopsy arrested homicide murdered	food	1.0	kitchen mouth glass food bread
worship	0.1	toil pray devote yourselves gather	sci-fi	0.0	suppose earth robots computer certain
moodiness	0.3	glumly snickered quizzically guiltily	fantasy	0.0	agreed magician dragon castle talent
informal	0.4	kinda damn heck guess shitty	military	0.1	ship captain lucky hour general

Table 1: Three high-precision (top) and three low-precision (bottom) descriptors for the **RMN** and **HTMM**, along with labels from an external evaluator and model precision (MP) computed via word intrusion experiments. The **RMN** is able to learn a variety of interpersonal states (e.g., love, sadness), while the **HTMM**'s most coherent topics are about concrete objects or events.

descriptor embeddings replaces topic-word probability. To control for randomness in the training process, we train three of each model, so the final experiment consists of 1,350 tasks ($K = 10, 30, 50$ descriptors per trial, three trials per model).

We collect judgments from ten different workers for each task using the Crowdflower platform.[8] Our evaluation metric, model precision (MP), is the fraction of workers that select the correct intruder word for a descriptor k. Low model precision signals descriptors that lack cohesive themes.

On average, the **RMN**'s descriptors are much more interpretable than those of the baselines, as it achieves a mean model precision of 0.73 (Figure 3) across all values of K. There is little difference between the model precision of the three topic model baselines, which hover around 0.5. There is also little difference between the **GRMN** and **RMN**; however, visualizing the learned character and book embeddings as in Figure 6 may be insightful for literary scholars. We show example high and low precision descriptors for the **RMN** and **HTMM** in Table 1; a full list is included in the supplementary material.

4.4 Do Trajectories Make Sense?

While the previous evaluation focused only on descriptor quality, our next experiment compares the trajectories learned by the best **RMN** model from the intrusion experiment (measured by highest mean model precision) to those learned by the best **HTMM** model, which is the only baseline capable of learning relationship trajectories. Workers read a plot summary and choose which model's trajectory best represents the relationship in question. We use the $K = 30$ setting because it provides the best balance between descriptor variety and trajectory interpretability.

For this evaluation, we crawl Wikipedia, Goodreads, and SparkNotes for plot summaries associated with our 1,383 books. We then remove all relationships where each involved character is not mentioned at least five times in the summary, which results in a final evaluation set of 125 relationships.[9] We present workers with two characters, a plot summary, and a visualization of trajectories learned by the **RMN** and the **HTMM** (Figure 4). The workers then select the trajectory that best matches the relationship described by the summary.

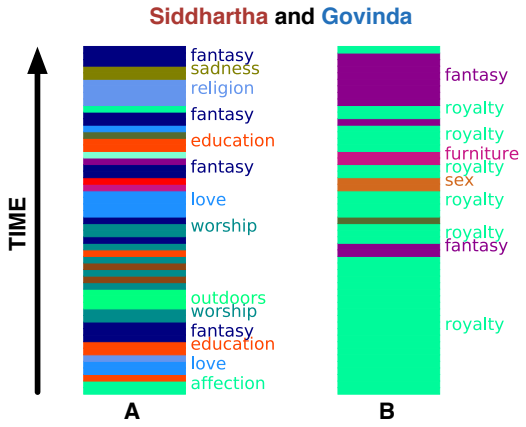
To generate the visualizations, we first have an external annotator label each descriptor from both models with a single word as in Table 1. For fairness, the annotator is unaware of the underlying models. For the **RMN**, we visualize trajectories by displaying the label of the argmax over descriptor weights $\boldsymbol{d}_t$ at each time step t. Similarly, for the **HTMM**, we display the most probable topic at each time step.[10]

The results of this task with seven workers per comparison favor the **RMN**: for 87 out of the 125 evaluated relationships (69.6%), the workers choose the **RMN**'s trajectory over the **HTMM**'s. We compute the Fleiss κ value (Fleiss, 1971) to correct our inter-annotator agreement for chance and find that

[8] http://www.crowdflower.com

[9] Without this filtering step, workers do not have enough information to compare the two models since most of the characters in our dataset are not mentioned in summaries.

[10] To reduce visual clutter, we ignore descriptors that persist for only a single time step.



Summary: Govinda is Siddhartha's best friend and sometimes his follower. Like Siddhartha, Govinda devotes his life to the quest for understanding and enlightenment. He leaves his village with Siddhartha to join the Samanas, then leaves the Samanas to follow Gotama. He searches for enlightenment independently of Siddhartha but persists in looking for teachers who can show him the way. In the end, he is able to achieve enlightenment only because of Siddhartha's love for him.

Figure 4: An example from the Crowdflower summary matching task; workers are asked to choose the trajectory (here, "A" is generated by the **RMN** and "B" by the **HTMM**) that best matches a provided summary that describes the relationship between Siddartha and Govinda (from *Siddartha* by Hesse).

$\kappa = 0.32$, indicating fair agreement among the workers. Furthermore, thirty-four relationships had unanimous agreement among the seven workers; of these, twenty-six were unanimous in favor of the **RMN** compared to only eight for the **HTMM**.

4.5 What Makes a Relationship Positive?

While the previous two experiments show that the **RMN** is more interpretable and accurate than baseline models, we have not yet shown that its insights can aid in drawing connections across various books and genres. As a first step in this direction, we investigate what makes a relationship positive or negative by comparing trajectories from the **RMN** and **HTMM** to static affinity annotations from a recently-released dataset (Massey et al., 2015) of fictional relationships. Expected correlations (e.g., murder and sadness are strongly negative descriptors) emerge alongside surprising ones (politics is negative, religion is positive).

The affinity labeling task of Massey et al. (2015) requires workers to describe a given relationship as positive, negative, or neutral. We consider only non-neutral relationships for which two annotators agree on the affinity label and remove all books not present in our own dataset. This filtering step results in 120 relationships, 78% of which are positive and the remaining 22% negative.

Model	Positive	Negative
RMN	education, love, religion, sex	politics, murder, sadness, royalty
HTMM	love, parental, business, outdoors	love, politics, violence, crime

Table 2: Descriptors most characteristic of positive and negative relationships, computed using existing annotations. Compared to the **RMN**, the **HTMM** struggles to coherently characterize negative relationships. Interestingly, both models show negative predispositions for political relationships, perhaps due to genre bias or class differences.

Since the annotations are static, we first aggregate our trajectories across all time steps. We compute K-dimensional "average positive" and "average negative" weight vectors $\boldsymbol{a}_p$ and $\boldsymbol{a}_n$ by averaging the relationship states $\boldsymbol{d}_t$ for the **RMN** and the document-topic distributions for the **HTMM** across all time steps for relationships labeled with a particular affinity. Then, we compute the vector difference $\boldsymbol{a}_p - \boldsymbol{a}_n$ and sort it to produce a ranked list of descriptors, where descriptors with positive differences occur more frequently in positive relationships. Table 2 shows the most positive and most negative descriptors; of particular interest is the large negative weight associated with political relationships from both models.

5 Qualitative Analysis

Our experiments show the superiority of the **RMN** over various topic model baselines in both descriptor interpretability and trajectory accuracy, but what causes the improved performance? In this section, we analyze similarities between the **RMN** and **HTMM** and look at qualitative examples where the **RMN** succeeds and fails. We also connect the findings of our affinity experiment to existing literary scholarship.

Both models are equally proficient at learning and assigning event-based descriptors (e.g., crime, violence, food). More specifically, the **RMN** and **HTMM** agree on environmental descriptions (e.g., boats, outdoors) and graphic sexual scenes (Figure 5, middle).

However, the **RMN** is more sophisticated with in-

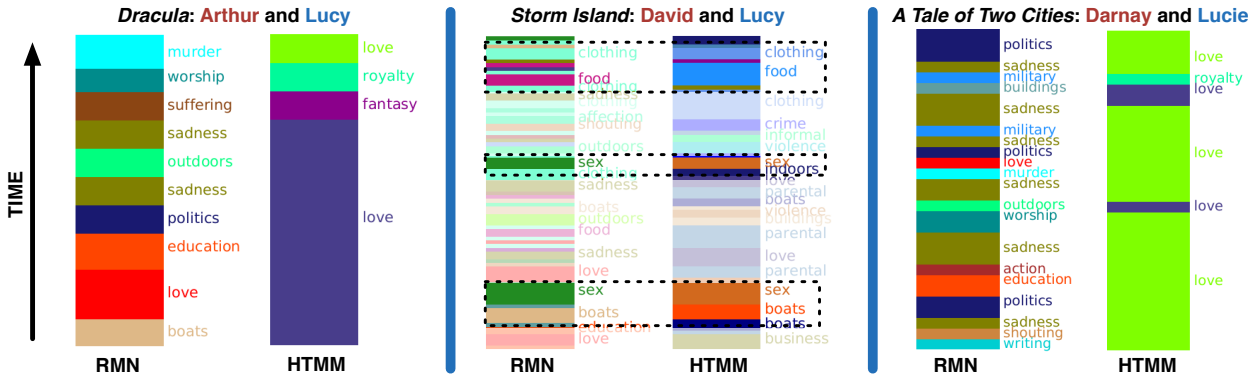


Figure 5: *Left*: the **RMN** is able to model Arthur and Lucy's trajectory reasonably well compared to our manually-created version in Figure 1. *Middle*: both models agree on event-based descriptors such as food and sex. *Right*: a failure case for the **RMN** in which it is unable to learn that Lucie Manette and Charles Darnay are in love.

terpersonal relationships. None of the topic model baselines learns negative emotional descriptors such as sadness or suffering, which explains the inaccurate **HTMM** trajectory of Arthur and Lucy in the left-most panel of Figure 5. All of the topic model baselines learn duplicate topics; in Table 2, one love descriptor is highly positive while a duplicate is strongly negative.[11] The **RMN** circumvents this problem with its uniqueness penalty (Equation 8).

While the increased descriptor variety is a positive, sometimes it leads the **RMN** astray. The model largely ignores the love between Charles Darnay and Lucie Manette in Dickens' *A Tale of Two Cities* due to book's sad tone; meanwhile, the **HTMM**'s trajectory, while vastly simplified, does pick up on the romance (Figure 5, right). While the **RMN**'s learnable book and character embeddings should help, the signal in a span cannot lead to the "proper" descriptor.

Both the **RMN** and **HTMM** learn that politics is strongly negative (Table 2). Existing scholarship supports this finding: Victorian-era authors, for example, are "obsessed with *otherness* . . . of antiquated social and legal institutions, and of autocratic and/or dictatorial abusive government" (Zarifopol-Johnston, 1995), while in science fiction, "dystopia——precisely because it is so much more common (than utopia)—— bears the aspect of lived experience" (Gordin et al., 2010). Our affinity data comes primarily from Victorian novels (e.g., by Dickens and George Eliot), leading us to believe that that the models are behaving reasonably. Finally, returning to the "extra" meaning of meals discussed in Section 1, food occurs *slightly* more frequently in positive relationships.

[11] This "duplicate love" phenomenon persists even when we reduce the number of topics.

6 Related Work

There are two major areas upon which our work builds: computational literary analysis and deep neural networks for natural language processing.

Most previous work in computational literary analysis has focused either on characters or events. In the former category, graphical models and classifiers have been proposed for learning character personas from novels (Bamman et al., 2014; Flekova and Gurevych, 2015) and film summaries (Bamman et al., 2013). The **NUBBI** model of Chang et al. (2009a) learns topics that statically describe characters and their relationships. Because these models lack temporal components (the focus of our task), we compare instead against the **HTMM** of Gruber et al. (2007).

Closest to our own work is the supervised structured prediction problem of Chaturvedi et al. (2016), in which features are designed to predict dynamic sequences of positive and negative interactions between two characters in plot summaries. Other research in this area includes social network construction from novels (Elson et al., 2010; Srivastava et al., 2016) and film (Krishnan and Eisenstein, 2015), as well as attempts to summarize and generate stories (Elsner, 2012).

While some of the relationship descriptors learned by our model are character-centric, others are more events-based, depicting actions rather than feelings;

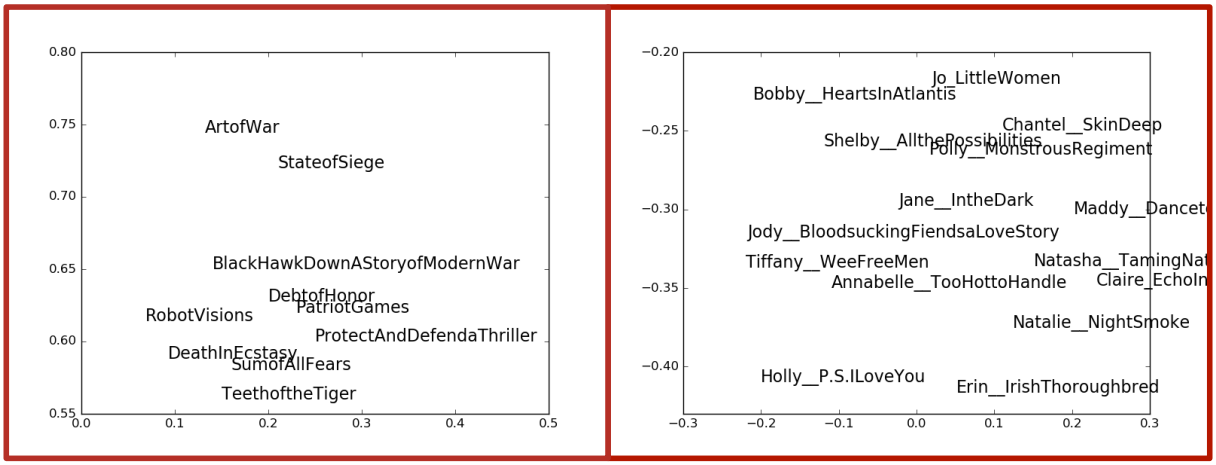


Figure 6: Clusters from PCA visualizations of the **RMN**'s learned book (left) and character (right) embeddings. We see a cluster of books about war and violence (many of which are authored by Tom Clancy) as well as a cluster of lead female characters from primarily romance novels. These visualizations show that the **RMN** can recover useful static representations of characters and books in addition to the dynamic relationship trajectories.

such descriptors have been the focus of much previous work (Schank and Abelson, 1977; Chambers and Jurafsky, 2008; Chambers and Jurafsky, 2009; Orr et al., 2014). Our model is more closely related to the plot units framework (Lehnert, 1981; Goyal et al., 2013), which annotates events with emotional states.

The **RMN** builds on deep recurrent autoencoders such as the hierarchical LSTM autoencoder of Li et al. (2015); however, it is more efficient because of the span-level vector averaging. It is also similar to recent neural topic model architectures (Cao et al., 2015; Das et al., 2015), although these models are limited to static document representations. We hope to apply the **RMN** to nonfictional datasets as well; in this vein, Iyyer et al. (2014) apply a neural network to sentences from nonfiction political books for ideology prediction.

More generally, topic models and related generative models are a central tool for understanding large corpora from science (Talley et al., 2011) to politics (Nguyen et al., 2014). We show representation learning models like **RMN** can be just as interpretable as LDA-based models. Other applications for which researchers have prioritized interpretable vector representations include text-to-vision mappings (Lazaridou et al., 2014) and word embeddings (Fyshe et al., 2015; Faruqui et al., 2015).

7 Conclusion

We formalize the task of unsupervised relationship modeling, which involves learning a set of relationship descriptors as well as a trajectory over these descriptors for each relationship in an input dataset. We present the **RMN**, a novel neural network architecture for this task that generates more interpretable descriptors and trajectories than topic model baselines. Finally, we show that the output of our model can lead to interesting insights when combined with annotations in an existing dataset.

Acknowledgments

We thank Jonathan Chang and Amit Gruber for providing baseline code, Thang Nguyen for helpful discussions about our model, and the anonymous reviewers for their insightful comments. This work was supported by NSF grant IIS-1320538. Boyd-Graber is also partially supported by NSF grants CCF-1409287 and NCSE-1422492. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsor.

References

Alekh Agarwal, Animashree Anandkumar, Prateek Jain, Praneeth Netrapalli, and Rashish Tandon. 2014. Learning sparsely used overcomplete dictionaries. In *Proceedings of Conference on Learning Theory*.

David Bamman, Brendan O'Connor, and Noah A. Smith. 2013. Learning latent personas of film characters. In *Proceedings of the Association for Computational Linguistics*.

David Bamman, Ted Underwood, and Noah A. Smith. 2014. A bayesian mixed effects model of literary character. In *Proceedings of the Association for Computational Linguistics*.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3.

Ziqiang Cao, Sujian Li, Yang Liu, Wenjie Li, and Heng Ji. 2015. A novel neural topic model and its supervised extension. In *Association for the Advancement of Artificial Intelligence*.

Nathanael Chambers and Daniel Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of the Association for Computational Linguistics*.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Association for Computational Linguistics*.

Jonathan Chang, Jordan Boyd-Graber, and David M Blei. 2009a. Connections between the lines: augmenting social networks with text. In *Knowledge Discovery and Data Mining*.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L Boyd-Graber, and David M Blei. 2009b. Reading tea leaves: How humans interpret topic models. In *Proceedings of Advances in Neural Information Processing Systems*.

Snigdha Chaturvedi, Shashank Srivastava, Hal Daumé III, and Chris Dyer. 2016. Modeling dynamic relationships between characters in literary novels. In *Association for the Advancement of Artificial Intelligence*.

J. Cognard-Black, M. Goldthwaite, and M. Nestle. 2014. *Books That Cook: The Making of a Literary Meal*. NYU Press.

Rajarshi Das, Manzil Zaheer, and Chris Dyer. 2015. Gaussian lda for topic models with word embeddings. In *Proceedings of the Association for Computational Linguistics*.

Michael Elad and Michal Aharon. 2006. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 15(12).

Micha Elsner. 2012. Character-based kernels for novelistic plot structure. In *Proceedings of the European Chapter of the Association for Computational Linguistics*.

David K Elson, Nicholas Dames, and Kathleen R McKeown. 2010. Extracting social networks from literary fiction. In *Proceedings of the Association for Computational Linguistics*.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah Smith. 2015. Sparse overcomplete word vector representations. In *Proceedings of the Association for Computational Linguistics*.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5).

Lucie Flekova and Iryna Gurevych. 2015. Personality profiling of fictional characters using sense-level links between lexical resources. In *Proceedings of Empirical Methods in Natural Language Processing*.

T.C. Foster. 2009. *How to Read Literature Like a Professor*. HarperCollins.

Alona Fyshe, Leila Wehbe, Partha P Talukdar, Brian Murphy, and Tom M Mitchell. 2015. A compositional and interpretable semantic space. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Michael D Gordin, Helen Tilley, and Gyan Prakash. 2010. *Utopia/dystopia: conditions of historical possibility*. Princeton University Press.

Amit Goyal, Ellen Riloff, and Hal Daumé III. 2013. A computational model for plot units. *Computational Intelligence Journal*, 29(3).

Amit Gruber, Yair Weiss, and Michal Rosen-Zvi. 2007. Hidden topic markov models. In *Proceedings of Artificial Intelligence and Statistics*.

Aapo Hyvärinen and Erkki Oja. 2000. Independent component analysis: algorithms and applications. *Neural networks*, 13(4).

Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, and Philip Resnik. 2014. Political ideology detection using recursive neural networks. In *Proceedings of the Association for Computational Linguistics*.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the Association for Computational Linguistics*.

Matt L. Jockers. 2013. *Macroanalysis: Digital Methods and Literary History*. Topics in the Digital Humanities. University of Illinois Press.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Vinodh Krishnan and Jacob Eisenstein. 2015. "You're Mr. Lebowski, I'm The Dude": Inducing address term formality in signed social networks. In *Conference*

of the North American Chapter of the Association for Computational Linguistics.

Angeliki Lazaridou, Elia Bruni, and Marco Baroni. 2014. Is this a wampimuk? cross-modal mapping between distributional semantics and the visual world. In *Proceedings of the Association for Computational Linguistics*.

Wendy G Lehnert. 1981. Plot units and narrative summarization. *Cognitive Science*, 5(4).

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the Association for Computational Linguistics*.

Philip Massey, Patrick Xia, David Bamman, and Noah A Smith. 2015. Annotating character relationships in literary texts. *arXiv:1512.00728*.

Viet-An Nguyen, Jordan Boyd-Graber, Philip Resnik, Deborah Cai, Jennifer Midberry, and Yuanxin Wang. 2014. Modeling topic control to detect influence in conversations using nonparametric topic models. *Machine Learning*, 95:381–421.

Bruno A Olshausen and David J Field. 1997. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23).

J Walker Orr, Prasad Tadepalli, Janardhan Rao Doppa, Xiaoli Fern, and Thomas G Dietterich. 2014. Learning scripts as hidden markov models. In *Association for the Advancement of Artificial Intelligence*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of Empirical Methods in Natural Language Processing*.

Roger Schank and Robert Abelson. 1977. *Scripts, Plans, Goals and Understanding: an Inquiry into Human Knowledge Structures*. L. Erlbaum.

Richard Socher, Quoc V Le, Christopher D Manning, and Andrew Y Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*.

Shashank Srivastava, Snigdha Chaturvedi, and Tom Mitchell. 2016. Inferring interpersonal relations in narrative summaries. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16.

Edmund M. Talley, David Newman, David Mimno, Bruce W. Herr, Hanna M. Wallach, Gully A. P. C. Burns, A. G. Miriam Leenders, and Andrew McCallum. 2011. Database of NIH grants using machine-learned categories and graphical clustering. *Nature Methods*, 8(6):443–444, May.

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 Empirical Methods in Natural Language Processing*.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *International Joint Conference on Artificial Intelligence*.

Ilinca Zarifopol-Johnston. 1995. *To kill a text: the dialogic fiction of Hugo, Dickens, and Zola*. University of Delaware Press.