Analyzing Fictional Relationships using Character Networks

Arushi Somani
arushisomani@berkeley.edu

Andrew Sue
gusue418@berkeley.edu

Christy Koh
christykoh@berkeley.edu

Krishnakumar Bhattaram
krishnabhattaram@berkeley.edu

Abstract

Existing approaches to generating character embeddings involve non-neural techniques, which are unable to capture nuanced interactions between characters and lack the ability to infer the relationship between characters who never had actual interactions. In contrast, we would like to predict embeddings for a single work or series of works, such that we can learn the sentiment and be able to infer relationships of any pair of characters. To the best of our knowledge, this has not been attempted before.

1 Introduction

A character network is a graph extracted from a story, with characters as nodes and interactions between them as edges. Since interactions between characters intuitively form the backbone of a plot, a number of analytical problems, such as character-level or interaction-level classification tasks, can be automatically applied using such a character network. One could ostensibly assess the validity of literary theories, produce graphical representations to visualize the strength and nature of relationships, summarize and simplify plots, or other educational applications. Furthermore, a character network can be viewed as a type of complex network, on which one could apply more complex topological analyses. There exist different methods of extracting character interactions, for example dialogue, direct actions, or a combination of both.

Most prior techniques rely on binary indicators of character interaction and character mentions. Furthermore, most prior works rely on non-neural methods to represent character interactions. For example, Naoya et al. attempt to cluster cross-fictional characters based on author, book, or textual similarity[3]. Our contributions in this paper are as follows:

- We introduce a novel sentiment-based method to create embeddings of fictional characters in a literary work.
- We outline a dynamic graph neural network based architecture that learns a mapping from dialog strings to a numerical sentiment measure.
- We train a model on the works of William Shakespeare and present an evaluation of this model

2 Related Works

Vincent Labatut and Xavier Bost examine research literature in character networks in their paper "Extraction and Analysis of Fictional Character Networks: A Survey".

They cite several case studies of insightful analyses run on character networks. For example, Grayson et al. [136] study the co-occurrence network of Austen's Sense and Sensibility. They find that the closest characters are also the wealthiest, a result that supports literary theory regarding social exclusivity in the social systems described in Austen's novels.

Labatut and Bost note that while there are many studies analyzing and extracting static networks, which describe characters' relationships over the course of a single work, there is a dearth of literature and tools to analyze dynamic networks that we expect to see in expanded works like TV or novel series. Inferring the dynamic behavior of character networks would be an essential step toward developing creative tools, such as for narrative generation. Accordingly, we are motivated to create a set of embeddings that capture this temporal, dynamic nature of character relationships.

Our chosen architecture is motivated by successes in recurrent architectures that capture the evolution of a social network. In "Feuding Families and Former Friends: Unsupervised Learning for Dynamic Fictional Relationships", Iyyer et al. propose a variant of a deep recurrent autoencoder that they call a "relationship modeling network" to learn character information across a corpus of different books. Their architecture leverages dictionary learning to learn a relationship descriptor and relationship trajectory for each edge. They learn a descriptor matrix R sized according to the number of desired topics, via a reconstruction task of an input span using a linear combination of the descriptors in R. The input span is a chronologically ordered series of dialogues between two characters, concatenated with the character information and the book title. Iyyer et al. find their method successfully and repeatably learns global descriptors that are both character-centric (ex: sadness, love, royalty) and event-based (ex: crime, violence, food).

Looking at the related problem space of social networks, we take inspiration from a paper by Twitter researchers Emanuele Rossi and Michael Bronstein. They propose an encoder neural network that ingests a stream of events to generate a time-dependent embedding for each node, i.e. individual, in the graph. They propose that their general encoder architecture can be trained with any decoder, for example, future interaction prediction.

Inspired by these studies, we attempt a similar recurrent architecture trained by the decoder-side task of sentiment prediction rather than the unsupervised reconstruction task.

3 Design and Implementation

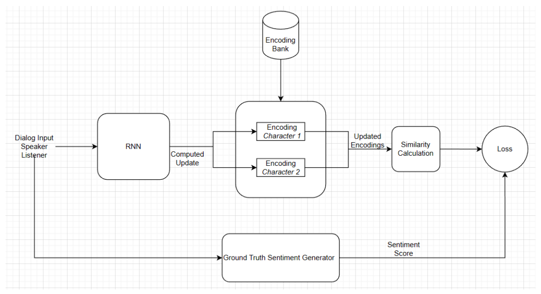
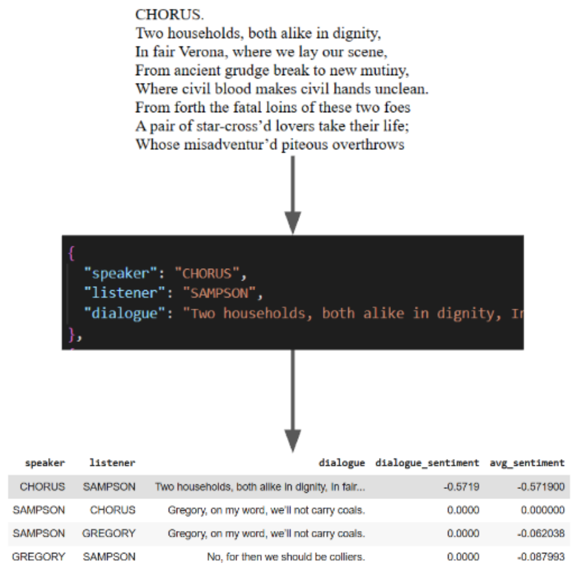


Figure 1: A Figure of our architecture.

Our approach to the problem is to design our own derivative of the canonical Graph Neural Network; we then pretrain it in the style of ChatGPT to compute the embeddings of the characters in Shakespeare's plays.

3.1 Data

Since we aim to find out relationships and sentiments between characters in a story, we used the dialogues in Shakespeare's plays as our training and testing data sets. Texts in the format of dialogues are especially suitable for our use as dialogues are great indicators of relationships between characters and are much easier to manipulate than narrative text. Dialogues, especially in plays, are usually said explicitly between characters, whereas there is a lot more ambiguity in monologues in many narrative texts.



CHORUS.
Two households, both alike in dignity,
In fair Verona, where we lay our scene,
From ancient grudge break to new mutiny,
Where civil blood makes civil hands unclean.
From forth the fatal loins of these two foes
A pair of star-cross'd lovers take their life;
Whose misadventur'd piteous overthrows

```
{
  "speaker": "CHORUS",
  "listener": "SAMPSON",
  "dialogue": "Two households, both alike in dignity, In
},
```

speaker	listener	dialogue	dialogue_sentiment	avg_sentiment
CHORUS	SAMPSON	Two households, both alike in dignity, In fair...	-0.5719	-0.571900
SAMPSON	CHORUS	Gregory, on my word, we'll not carry coals.	0.0000	0.000000
SAMPSON	GREGORY	Gregory, on my word, we'll not carry coals.	0.0000	-0.062038
GREGORY	SAMPSON	No, for then we should be colliers.	0.0000	-0.087993

Figure 2: Our data pipeline is where we extract dialogues, deduce the speaker and listener and compute the sentiment of the dialogue.

3.1.1 Dialogue Extraction

From each manuscript, we separated the spoken dialogue from other labels and stage directions. To scope down the task, each piece of dialogue was treated as acting between a speaker and a single listener and labeled with both characters. To determine the speaker and listener of each dialogue, we used a reference list of characters in the play to associate each line of dialogue with its speaker. Assuming sequential pairwise conversation, we labeled the listener as the previous speaker.

3.2 Sentiment Calculations

From the dialogues extracted, we used VADER (1), a rule-based sentiment analysis tool, to find the sentiment of each specific dialogue. The sentiments are a scalar value between -1 and 1. Negative values mean negative sentiments and positive values mean positive sentiments. Each dialogue with its respective speaker, listener, and sentiment is stored in a table. Furthermore, we also computed the average sentiments between pairs of characters.

3.3 Architecture

Taking inspiration from previous works, such as the Temporal Graph Network proposed by Twitter, we designed a custom architecture to generate character embedding that encompasses the relationships between characters with dialogues between characters as input. A diagram of our architecture is shown in 1.

Each data point is a dialogue between two characters. A forward pass of our network is the following:

1. Pass the dialogue through BERT to get a text embedding, $e_{dialogue}$.
2. We take the embedding involved with the two characters $v_{speaker}$, $v_{listener}$ out from the embedding bank, update the embeddings based on the results of two MLPs, $f_{speaker}$, $f_{listener}$
3. Pass the updated embeddings $v'_{speaker}$, $v'_{listener}$ through another MLP, p, to get the sentiment between the two characters.
4. Calculate the loss of the sentiment from our network and backpropagate.

Our network can also be visualized similarly to a Dynamic Graph Network. In essence, we are training a graph network that only updates one edge at a time, dictated by the following update rules and node values where e are the messages being passed, v are the embeddings or the node value of the characters, f are the update functions for the characters, p is the function that computes sentiment and S is the final calculated sentiments:

$$e_{dialogue} = E(Dialogue) \tag{1}$$

$$\Delta v_{speaker} = f_{speaker}(v_{speaker}, v_{listener}, e_{dialogue}) \tag{2}$$

$$\Delta v_{listener} = f_{listener}(v_{speaker}, v_{listener}, e_{dialogue}) \tag{3}$$

$$v'_{speaker} = v_{speaker} + \Delta v_{speaker} \tag{4}$$

$$v'_{listener} = v_{listener} + \Delta v_{listener} \tag{5}$$

$$S = p(v'_{speaker}, v'_{listener}) \tag{6}$$

The Graph Neural Network representation of our architecture is shown in 3

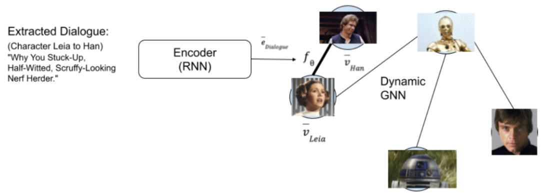


Figure 3: Our architecture can also be envisioned as the following GNN with characters as nodes, the dialogue sentiments and character relationships as the edges.

4 Evaluation

Through our evaluation, we attempt to answer the following research questions:

- **RQ1**: Do the learned character embeddings accurately represent the relationships between characters?
- **RQ2**: Does the learned network predict the sentiment of future interactions between characters well?

4.1 RQ1: Embedding Closeness

First, we empirically verify our embedding representation by visualizing the closeness of our embeddings. We use four-dimensional embeddings to represent each character, and visualize them on a 2D space using T-SNE (Figure 4).

We examine character closeness, and determine that characters that interact positively and frequently are closer than characters that interact negatively or infrequently. A few examples can be found in Appendix A.

For evaluation, we manually create clusters of characters who are aligned in the play. For example: Juliet and Capulet exist in the same manual cluster, while Romeo and Montague exist in another manual cluster. We make the following assumption: If the average distance between elements in the same cluster are less than the average distance between elements across clusters, our embeddings are well-aligned. We demonstrate the alignment of our embeddings below.

	Romeo	Juliet	Neutral
Romeo	6.634e-2	8.586e-2	10.942e-2
Juliet	8.857e-2	9.076e-2	11.857e-2

Table 2: Average Distance between Cluster Embeddings

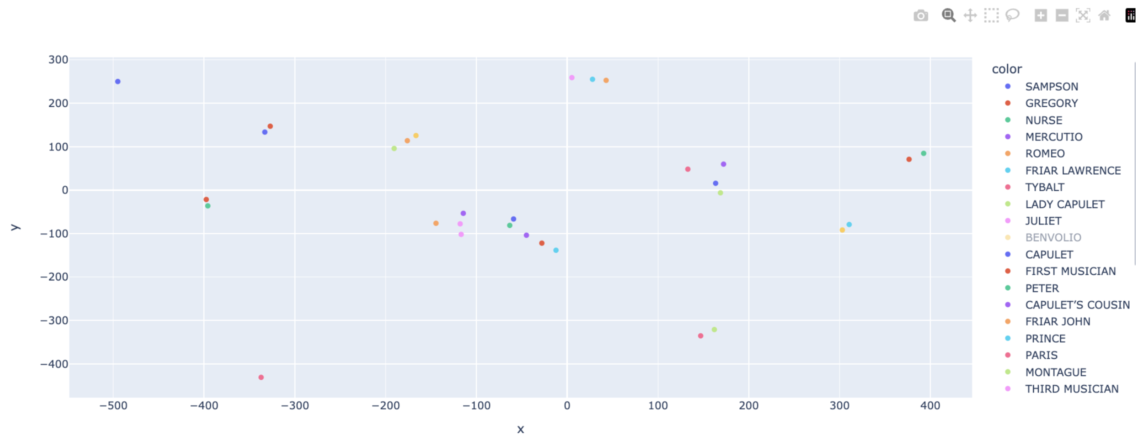


Figure 4: Embedding Visualization for Romeo & Juliet

4.2 RQ2: Sentiment Prediction

We also use our learned character embeddings and network to predict the sentiment of future dialogues. We compare this performance to that of a simplified system without character embeddings — this system simply has BERT with a linear layer for prediction. Our test set comprises dialogue from the latter half of the play, while our training set is the former half. This is to maintain the temporal nature of dialogue.

We include a plot of batched loss against epochs of training below. We note the noisiness of our loss curve. We hypothesize that the reason for this is that the embedding update function creates gradient explosions, which causes some training runs to not convergence. This is a limitation of our design, and something we look forward to investigating in future work.

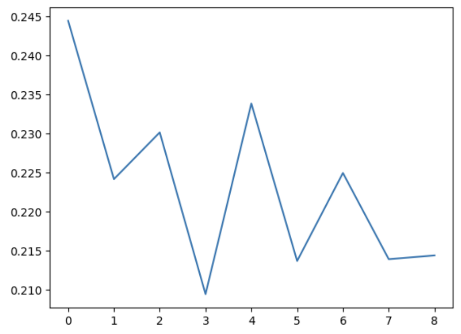


Figure 5: Batch Loss against Epochs of Training

The simplified BERT system has a test accuracy of 74%, while our embeddings model test a test accuracy of 63%. We note that the simplified system performs better than the one with character embeddings. We hypothesize that this is because the embeddings are a lossy representation of dialog. It is an attempt to condense N dialogs into K embeddings, where $N >> K$. Different interactions pull character embeddings in different directions, and thus it is worse at predicting sentiment than a linear layer. Future work should investigate how to maintain sentiment analysis accuracy with embedding accuracy.

We also attempt to quantitatively measure our predictions. On our test set, we note that approximately 81% of data pints are within 0.8 in mean squared error, and approximately 97% of datapoints are within 1 in mean squared error. Intuitively, this means that most data points are categorized as more correct than incorrect, which would be if the MSE was greater than 1.

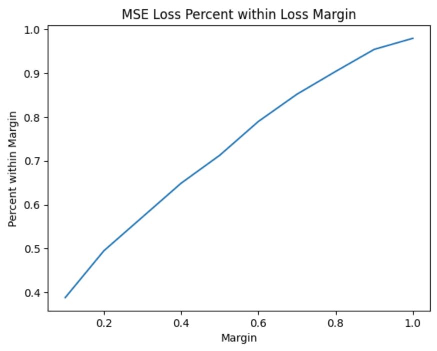


Figure 6: Percentage of test points within margin euclidean distance of ground truth

5 Limitations and Future Work

5.1 Impurity of Ground Truth

While we used VADER to generate our ground truth sentiment values, we note that VADER marks many dialogues at a 0.0 sentiment (neither positive, nor negative). This handicaps training and restricts gradients. Future work should investigate better generation of ground truth labels.

5.2 Training Divergence

We noticed gradient explosion and divergence during the training process. While the embeddings are still good representations of the characters, we note that because of this, the prediction remains poor, doing only slightly better than average.

5.3 Training Procedure

Our neural network is pretrained on a single dialogue per iteration, with loss immediately propagated back through the linear head linked to an MSE loss compared to the sentiment of the dialogue. This has a few issues:

The network has already seen the dialogue and conducted its update before the linear layer does its prediction. This means that the model may be likely to learn to ignore the embeddings and simply conduct inference based on the BERT dialogue vector. Furthermore, due to a limited amount of data, we conduct the backprop after exposure to every dialogue. This leads to potential in-place modification issues for the RNN-analogous updates sequence. For this reason, we do not allow gradients to flow through multiple layers. Instead, we take each input character embedding as a network input.

We plan to address these issues by augmenting our story dataset with dialogue sequences that mask certain dialogues, and using the loss on the entire set of sentiment outputs resulting from the linear heads on the batch of masked dialogue sequences. This BERT-style pretraining allows gradient propagation between layer, which should reduce the noise of the training procedure, and forces the embeddings to be active. We also plan to fine-tune our network parameters on story-level edge and node features such as character gender and relationships.

6 Conclusion

We demonstrated that it is possible to learn sentiment-based fictional character embeddings from limited data — here, a singular play. These embeddings demonstrated quantitatively and qualitatively that they accurately represent the relationships between the characters. This technique can be extended to create embeddings for analyzing episodic works of fiction, to different mediums, as well as across works of fiction.

References

Hutto, Clayton, and Eric Gilbert. "Vader: A parsimonious rule-based model for sentiment analysis of social media text." Proceedings of the international AAAI conference on web and social media. Vol. 8. No. 1. 2014.

[1] Lee, J., Jung, J. J., & Kim, T. (2020). Learning Hierarchical Representations of Stories by Using Multi-Layered Structures in Narrative Multimedia. Sensors (Basel, Switzerland), 20(7). https://doi.org/10.3390/s20071978

[2] O-Joun Lee, Jason J. Jung (2020). Story embedding: Learning distributed representations of stories based on character networks, Artificial Intelligence, Volume 281, 103235, ISSN 0004-3702, https://doi.org/10.1016/j.artint.2020.103235.

[3] Naoya Inoue, Charuta Pethe, Allen Kim, and Steven Skiena. 2022. Learning and Evaluating Character Representations in Novels. In Findings of the Association for Computational Linguistics: ACL 2022, pages 1008–1019, Dublin, Ireland. Association for Computational Linguistics. [4] Deep learning on dynamic graphs; Emanuele Rossi, https://blog.twitter.com/engineering/en$_us/topics/insights/2021/temporal - graph - networks$

Emanuele Rossi & Ben Chamberlain & Fabrizio Frasca & Davide Eynard & Federico Monti & Michael Bronstein. "Temporal Graph Networks for Deep Learning on Dynamic Graphs". 2020.

Appendix

Appendix A: Examples of Embedding Closeness

We highlight a few character embedding relationships, as examples of embeddings learning relationships/

Romeo, Juliet and Tybalt

Note the closeness of Romeo and Juliet's embeddings. Tybalt is the primary antagonist of Romeo and Juliet — note his distance from both Romeo and Juliet.

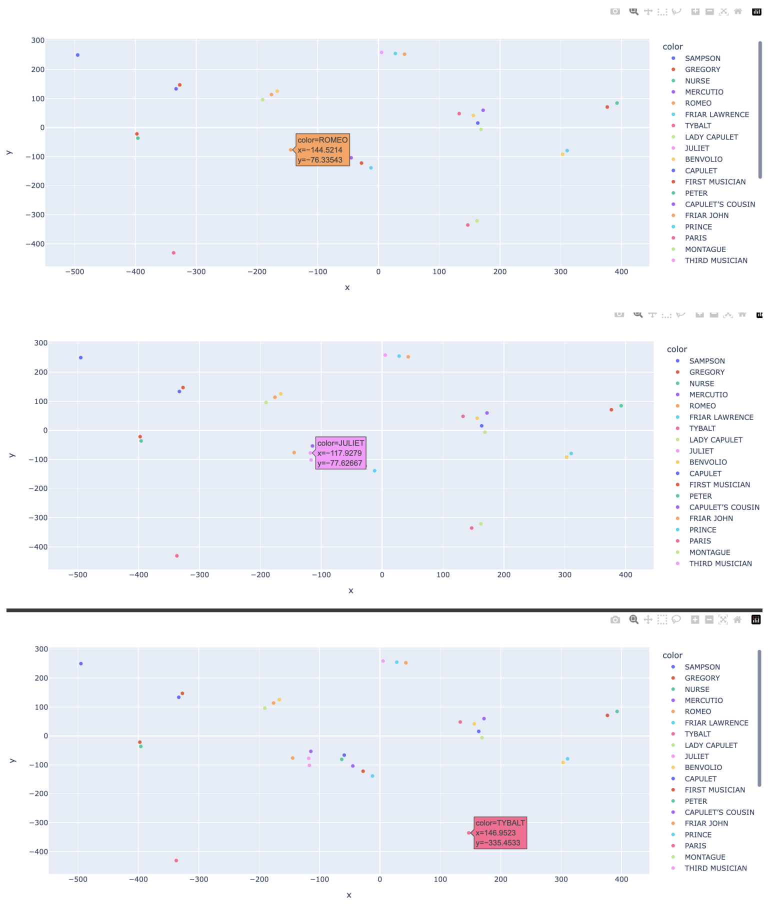


Figure 7: Romeo, Juliet and Tylbalt's Embeddings