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

Character embeddings are used as tools for literary analysis to compare narratives and analyze character psychology. Often, these techniques are transferred to real-world settings and applied on social networks. Existing approaches to generating character embeddings involve non-neural approaches, 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 to infer relationships between any pair of characters. To the best of our knowledge, this has not been attempted before. Our work is publicly available here.

Part I

Reviewer Comments

1 Reviewer 1

1. **Literature Background/Problem Formulation**

   The author wants to predict relationship between characters in literature works. There are several previous works, like the survey conducted by Vincent Labatut and Xavier Bost, which cited several study on character network, and shows that this technique helps in literature analysis. *Excellent work, no actions needed.*

2. **Technical Approach/Deep Learning Techniques**

   The author does site several papers or model names to show where their design of model come from. However, it's better to explain them more specifically in your paper, because the readers don't know what 'Dynamic Graph Network', 'BERT' and 'the style of ChatGPT' mean. *Small improvement needed*

3. **Experiments and Contributions**

   The author evaluate Embedding Closeness and Sentiment prediction. By plot each character in the embedding space, we can see how 'close' they are to each other. In sentiment prediction part, the author compare embedding model with simplified BERT. There is also analysis about why simplified BERT outperforms embedding model. *Excellent work, no actions needed.*

4. **Code and Datasets**

   The dataset are dialogues extracted from Shakespeare's work. The code can run and experiments are reproducible. *Excellent work, no actions needed.*

5. **Readability/Clarity**

   Clarity needs to be improved. *Small improvement needed*

2 Reviewer 2

1. **Literature Background/Problem Formulation**

   The problem is clearly stated and relevant literature cited.*Excellent work, no actions needed.*

2. **Technical Approach/Deep Learning Techniques**

   A citation/brief explanation of BERT could be helpful as background. *Excellent work, no actions needed.*

3. **Experiments and Contributions**

   Their baseline model performs better. Authors could state some ideas for how the embedding model accuracy could be improved, or if using embeddings confers other advantages besides accuracy. A few systemic experiments to justify hyperparameters/choices in the deep learning architecture would be helpful; does this affect the accuracy? Figure 5A is missing axis labels. In Figure 4, the colors repeat so the reader can't distinguish certain characters; it might be better to use a different color for every data point. It is unclear what 'neutral' refers to in Table 2. *Small improvement needed*

4. **Code and Datasets**

   The provided .ipynb ran in Google Colab and the code is included in the Github repo. *Excellent work, no actions needed.*

5. **Readability/Clarity**

   The paper is well organized. There are a few typos ("data pints") and grammatical errors. The citations could be better organized (e.g. Labatut and Bost in the Related Works section was not cited the References; Grayson et al. have a reference number that does not exist in References). *Small improvement needed*

3 Reviewer 3

1. **Literature Background/Problem Formulation**

   The literature review appears comprehensive, and it sets a reasonable basis for where/why the authors may seek to improve upon previous findings. The two unclear points, for me, were the definition of character relationships and the definition of a dynamic network. For character relationships, I spent the first half or so of the paper thinking the goal was to predict relationships such as parent/child, enemy, friend, etc, whereas later reading led me to believe the relationship we intended to predict instead lay purely on a positive/negative scale. I think a clarification of this might be helpful. For the definition of a dynamic network, it's somewhat clear what the benefit of updating the relationships over time could be, but it may be nice to see more scaffolding as to what those updates could actually look like in practice. Do they happen after every episode of a TV series, or do they happen every sentence in a work? The paper makes me lean toward the latter, but I'm still not fully sure. *Small improvement needed.*

2. **Technical Approach/Deep Learning Techniques**

   I liked this part! The approach taken seemed clear to me and accurately reflected in the code. I like the use of some in-class materials (like BERT), and some less emphasized in class (like VADER) *Excellent work, no actions needed.*

3. **Experiments and Contributions**

   In the "dialogue extraction" section, do you have a justification for making the previous speaker the listener of the next speaker's line? In large group scenes, or even scenes in which trios are deliberating, why can't a piece of dialogue be duplicated and used to train more than 1 relationship? . Also, the figure you provide makes it look like you did in fact do this duplication, which feels somewhat like it contradicts the description provided.

   The discussion as to the shortcomings of the experiment and potential further work is quite good! For the comment that the fact that losses are not batched and vary by character and dialogue, have you tried any forms of batching to reduce this noise? For the fact that VADER sets the sentiment of many dialogues to zero, leading to vanishing gradients, have you tried any kind of shifting/bias/etc to mitigate this? *Small improvement needed*

4. **Code and Datasets**

   The code is clear and consistent with the datasets, but the results I get are not consistent (specifically, the MSE loss plot is logarithmic with a linear end, which isn't like the linear plot included in the report, and the embedding graph is dramatically different). It also took about 30 minutes to execute the training loop on colab, and it might be nice to have some kind of warning / informative bit of text to let users know that before attempting to replicate your results. In the code, the loss plots should also have labels – it's hard to tell what's what, when loss is getting plotted, etc. *Medium improvement needed.*

5. **Readability/Clarity**

   The paper is good! One typo I saw: "which causes some training runs to not convergence", in section 4.2 *Small improvement needed*

Part II

Responses to Review Comments

4 Response to Reviewer 1

We thank the reviewer for their review! We have made the following changes in response:

- In the "Design and Implementation" section, we elaborate on dynamic graph networks, our use of BERT, and add an algorithm that walks through a single iteration of the forward pass of the architecture.
- To improve clarity, we attempt to explain at depth our architecture and update our visualizations for the same. We also attempt to provide more intuition into the domain of problems that character networks tackle.

We hope that these amendments help minimize the problems the reviewer noticed in our work!

5 Response to Reviewer 2

We thank the reviewer for their feedback! We have made the following changes in response:

- We cite and briefly explain BERT in our "design and implementation" section.
- As per the reviewer's advice, we highlight how our sentiment analysis test accuracy and embedding accuracy might not be proxies of each other.
- We add a small evaluation of hyperparameters against accuracy measures as justification for our hyper-parameter choices. We elaborate further on our comparison to our baseline of BERT with an MLP predictor to analyze the value of our architectural choices. We mention that we would enjoy investigating and comparing more architectures in future work.
- We update Figure 5A to have correctly labeled axes, and ensure that all other figures in the paper have so as well. We point readers towards Appendix A, which hopefully helps make Figure 4 more usable. We add a definition for "neutral" as mentioned in Table 2.
- We fix all mentioned typos and update our bibliography to be comprehensive.

We hope our amendments help correct the problems highlighted in the review!

6 Response to Reviewer 3

We thank the reviewer for their feedback! We have made the following changes in response:

- We clarify in the introduction section further what we mean by Character Networks, and outline the problem that our approach solves more clearly in the Design and Implementation section. We clarify that we are predicting scalar sentiment values. We add an algorithm section to clarify what a single forward pass through our network looks like. We add clarification for what the edge updates on a dynamic graph look like through the algorithm as well.
- We highlight the flaw in our dialogue extraction methodology as mentioned, and highlight the need for future work to handle for many-to-one conversations. We update our descriptions to ensure that the description and visuals provided do not seem to conflict each other. We add a mention regarding batching and highlight minimal improvements shown through batching. We add a section highlighting smoothing of ground truth labels to mitigate VADER's flaws of setting dialogue sentiment to zero.
- We update our code to use a generator with a manually set seed to ensure deterministic results. We add a warning highlighting the long duration taken to run the notebook. We add labels and titles to all plots in the notebook.

- We remove the typo from section 4.2.

We hope that these updates serve to minimize the problems highlighted in the paper.

We once again thank the reviewers for their helpful comments!

Part III

Final Submission

7 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. An edge exists between two characters if they interact in a given scene, and doesn't if they don't. (1) Furthermore, most prior works rely on non-neural methods to represent character interactions. (1) For example, Naoya et al. (3) attempt to cluster cross-fictional characters based on author, book, or textual similarity. 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. The similarity between two characters embeddings represents the positivity of the relationship between the characters.
- We outline a dynamic graph neural network based architecture that learns a mapping from dialog tokens to a one-dimensional numerical sentiment measure.
- We train a model on the works of William Shakespeare, BARDNET, and present an evaluation of this model.

8 Related Works

Labatut and Bost (1) examine literature in character networks in "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. (9) 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.

They further 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 that analyze dynamic networks which would more accurately model expanded works like TV or novel series. Inferring the dynamic behavior of character networks would be essential for creative tools, such as in 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. (5) 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. They 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).

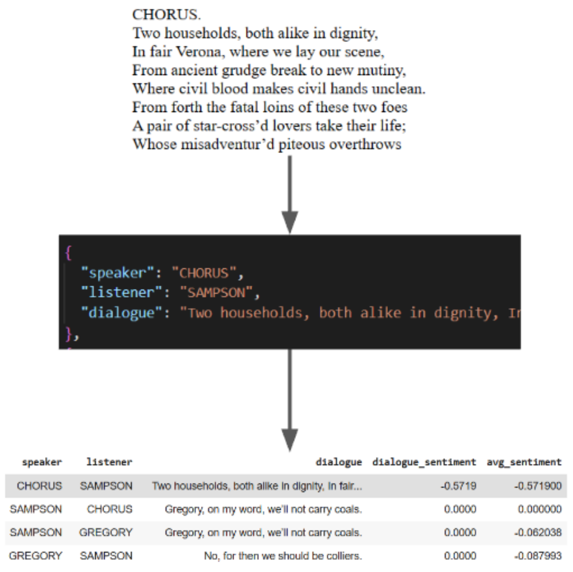
Looking at the related problem space of social networks, we take inspiration from a paper by Twitter researchers Emanuele Rossi and Michael Bronstein (7). They propose an encoder neural network that ingests a stream of events to generate a time-dependent embedding for each node, i.e. individual, in the 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.

9 Design and Implementation

9.1 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 they 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, I
},
```

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

9.1.1 Dialogue Extraction

Each piece of dialogue was treated as acting between a speaker and a single listener and labeled with both characters. Assuming sequential pairwise conversation, we labeled the listener as the previous speaker. Future work could expand the scope of extraction to many-to-one conversations. We use BERT (8) to create embeddings of the dialogues. BERT is a "bi-directional encoder" that helps encode vector representations of language — here, dialogues.

9.1.2 Sentiment Calculations

From the dialogues extracted, we used VADER (2), 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 correspond to negative sentiments and positive values to positive sentiments. We used label smoothing on the VADER values, which were often extreme values of -1, 1, and 0.

10 BARDNET Architecture

Taking inspiration from previous works, such as the Temporal Graph Network proposed by Twitter, (7) we designed a custom architecture. We use a Dynamic Graph Neural Net for our problem. These are a type of neural network that operates on time-varying graphs. Nodes in the network represent individuals, and the edges represents their interactions. For each interaction, the node embeddings are updated. A diagram of our architecture is shown in 2.

```
Algorithm 1: Forward Pass: BARDNET
Data: Literary work $L$, Dialog $D$, Speaker $s$, Listener $l$, for one dialog in work $L$
Result: Sentiment of Dialogue $S$

Fetch Embeddings
    $e_D \leftarrow \text{BERTEmbed}(D)$
    $v_s, v_l \leftarrow \text{GetCharEmbed}(s),\ \text{GetCharEmbed}(l)$
Perform GNN Update
    $\Delta v_s \leftarrow \text{GNNUpdate}_{speaker}(v_s, v_l, e_D)$
    $\Delta v_l \leftarrow \text{GNNUpdate}_{listener}(v_s, v_l, e_D)$
    $v'_s \leftarrow v_s + \Delta v_s$
    $v'_l \leftarrow v_l + \Delta v_l$
Convert Character Embedding to Scalar Sentiment
    $S \leftarrow \text{MLP}(v'_s, v'_l)$
```

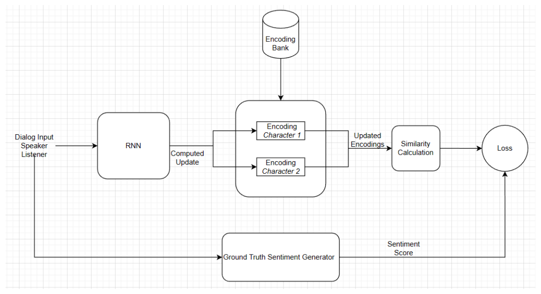


Figure 2: A Figure of our BARDNet architecture.

11 Evaluation

Through our evaluation, we attempt to answer the following 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?

11.1 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, Bidirectional Encoder Representationsand 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.

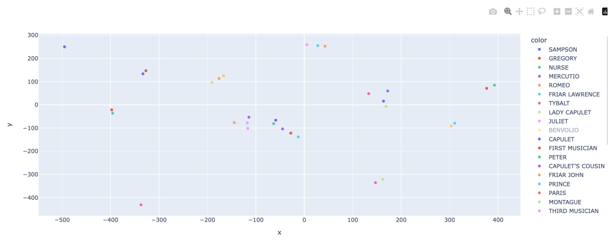


Figure 3: Embedding Visualization for Romeo & Juliet, further details in Appendix A

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. Characters like the Apothecary serve as neutral characters. 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

11.2 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 diverge. Moreover, the unbatched loss is extremely noisy since each loss is for different characters, and different dialogues. **This is the primary limitation of our design**, and something we look forward to improving in future work.

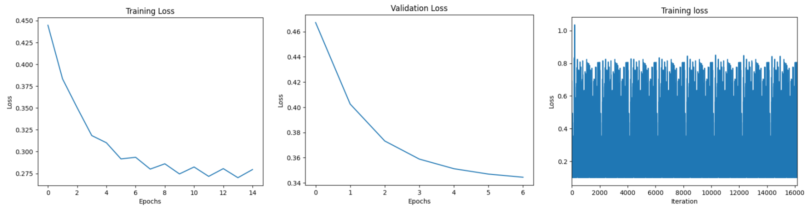


Figure 4: Figure A: Batch Training Loss against Epochs of Training. Figure B: Batch Validation Loss against Number of Dialogues over one epoch. Figure C: Unbatched Training Loss per Dialog approach epochs

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. However, **the accuracy of our embeddings suggest that correct embeddings and accuracy on sentiment analysis in the test set are imperfect proxies of each other.** That is to say, despite our low accuracy, our character embeddings are demonstrably good. Future work should investigate how to maintain sentiment analysis accuracy with embedding accuracy.

In Appendix B, we attempt to measure the correctness of our approach using the test MSE loss. We note that the model predicts 97% of the correct scalar sentiment value within an error margin of 0.8.

11.3 Hyperparameter Search

We perform basic grid search to measure loss and accuracy against various hyperparameters. We vary `edge_dim`, and `character_embedding_dim`, the dimensions of the edges of the GNN and the node character embeddings respectively. We hold other hyperparameters, like epochs and the bert embedding dimension, constant. The results demonstrate that the optimal values are 4 for the character embedding dimension and 16 for the edge dimension. The results are in the appendix

12 Limitations and Extensions

12.1 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.

12.2 Training Divergence

Despite implementing adaptations such as gradient clipping, we occasionally observed gradient explosion and divergence at different random seeds of 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.

12.3 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.

13 BERT-Style Pretraining

In a subsequent experiment, we attempt pretraining our model on sentiment prediction task before running inference on relationship analysis. In this case, we test against the downstream task of familial relation prediction. We attempt training BardNet by masking, as well as a new Transformer architecture.

13.1 Pretraining Procedure

Our pretraining procedure takes inspiration from BERT-style pretraining, and is centered around a masking procedure for a sequential input. A single input is the dialogue sequence comprising all the interactions in a single story. We choose a random percentage of dialogues to replace with a mask token <MASK> which is constant and preset. A masked dialogue retains the speaker and listener attributes but loses the dialogue embedding. We then use the sentiments on all the dialogues as the pretraining objective, with loss comparing a linear head output based on the per-dialogue state.

13.2 Vanishing and Exploding Gradients

During the training process, we realize that gradients seems to explode after any sequence longer than 20 dialogues. This is a common issue with RNNS, which often suffer from either exploding or vanishing gradients. introduce a modification inspired by the Truncated Backpropagation Through Time (BPTT) algorithm (10). In this algorithm, we detach gradients every 20 dialogues to avoid an excessively deep propagation of gradients. The loss curve, pictured in the appendix, was not satisfactorily converging for our pretraining task. We also tried gradient clipping and layer norm

13.3 Transformer

Due to poor model convergence even on the pretraining task, we decided to pre-train a transformer on the dialogue sentiments and then finetune the transformer to predict relationships between two arbitrary characters in the same story. The advantage of using a transformer is that it will not be limited by a temporal window as it is ingesting the entire story. We implemented a naive transformer that includes 2 linear layers with 2 encoder layers with single-head attention to provide the dialogue encodings that will be used in the finetuning task downstream.

For the pre-training stage, we use the same pre-training procedure for BARDNet.

13.3.1 Fine-tuning

For the fine-tuning procedure, we first generate the "ground truth" relationship labels. the labels are organized in a dictionary where the key is the pairwise name of the characters in question, and the value is a vector that one hot encodes the relationships between the two characters. The vector is 9 dimensional, describing: Friendship/Alliance, Love/Attraction, Hatred/Enmity, Family Relation, Servitude, Rivalry, Disrespect and Neutral relationships. One would denote such a relationship exists and a zero represents such a relationship is nonexistent. The ground truth relationship is manually constructed referencing a character network defined in the following link (https://www.rsc.org.uk/shakespeare-learning-zone/romeo-and-juliet/character/relationships).

The MLP that predicts the character relations is 2 layers deep with tanh and ReLU activations. This predictor replaced the original linear layer that predicts the sentiments in the transformer.

During fine-tuning, a "dialogue" of the masked token concatenated with the speakers and the listener is inputted and is then back-propagated with MSE Loss with the relationship vector between the corresponding speaker and listener pair. As the weights inside the transformer are frozen, we hope the memory of the transformer will construct an embedding that incorporates the relationship between two characters which can be decoded by an MLP.

The result can be found in the appendix.

Occasionally, we noticed that the testing loss (shown in figure 6)d for the relationship predictor is often better than the training loss, which at first glance seems counterintuitive. However, we believe this is due to the imbalance of labels during our relationship ground truth generation. For characters that are not in the relation graph in our reference, we assume the characters has no relations with each other. Thus in the testing data, and training data, there is a high proportion of labels that includes vectors that are all zeros, so by guessing a zero vector, the network has a high chance that it can get a low loss in the testing data, which by chance might be lower than the training data. Furthermore, we only have 199 relationship pairs, this small number of data points leads to high variance.

14 Conclusion

We demonstrated that it is possible to learn sentiment-based dialogue 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 could be extended beyond a sentiment-based downstream task to do further work analyzing episodic works of fiction, to different mediums, as well as across works of fiction.

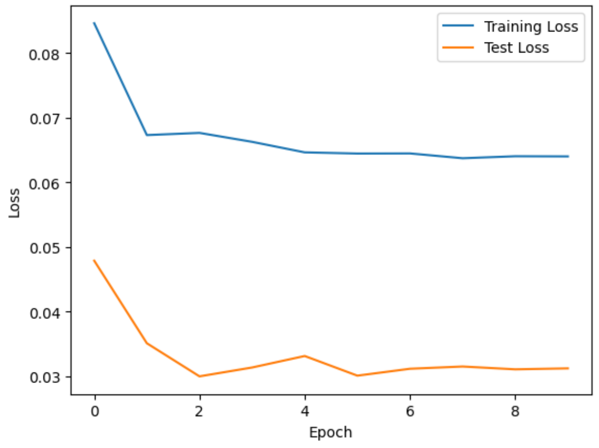


Figure 5: The fine-tuned loss on the relationship predictor.

References

[1] V. Labatut and X. Bost. 2019. "Extraction and Analysis of Fictional Character Networks: A Survey." ACM Computing Surveys 52(5):89. https://doi.org/10.1145/3344548

[2] C. Hutto and E. Gilbert. 2014. "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.

[3] J. Lee, J. J. Jung, and T. Kim. 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

[4] O. Lee and J. 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.

[5] M. Iyyer, A. Guha, S. Chaturvedi, J. Boyd-Graber, and H. Daumé. 2016. Feuding families and former friends: Unsupervised learning for dynamic fictional relationships. In Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 1534–1544. https://doi.org/10.18653/v1/N16-1180

[6] 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. https://aclanthology.org/2022.findings-acl.81.pdf

[7] E. Rossi. B. Chamberlain, F. Frasca, D. Eynard, F. Monti, M. Bronstein. "Temporal Graph Networks for Deep Learning on Dynamic Graphs". 2020. https://blog.twitter.com/engineering/en_us/topics/insights/2021/temporal-graph-networks

[8] J. Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” ArXiv abs/1810.04805 (2019)

[9] S. Grayson, K. Wade, G. Meaney, and D. Greene. 2016. The Sense and Sensibility of Different Sliding Windows in Constructing Co-occurrence Networks from Literature. In International Workshop on Computational History and Data-Driven Humanities. 65–77. https://doi.org/10.1007/978-3-319-46224-0_7

[10] Mozer, M. C. (1995). "A Focused Backpropagation Algorithm for Temporal Pattern Recognition". In Chauvin, Y.; Rumelhart, D. (eds.). Backpropagation: Theory, architectures, and applications. ResearchGate. Hillsdale, NJ: Lawrence Erlbaum Associates. pp. 137–169.

Appendix

Appendix A: Examples of Embedding Closeness

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

Embeddings of Characters that Interact

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. Moreover, note the closeness of Gregory and Sampson's embeddings.

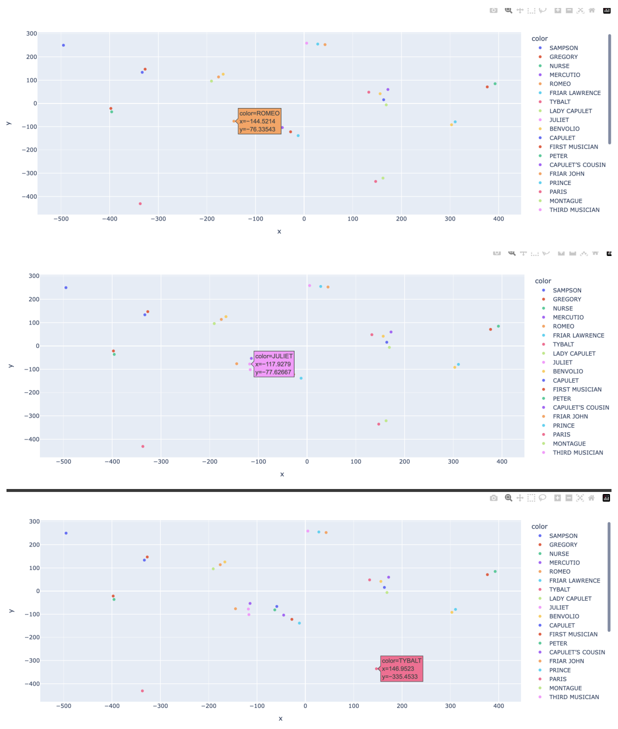


Figure 6: Romeo, Juliet and Tylbalt's Embeddings

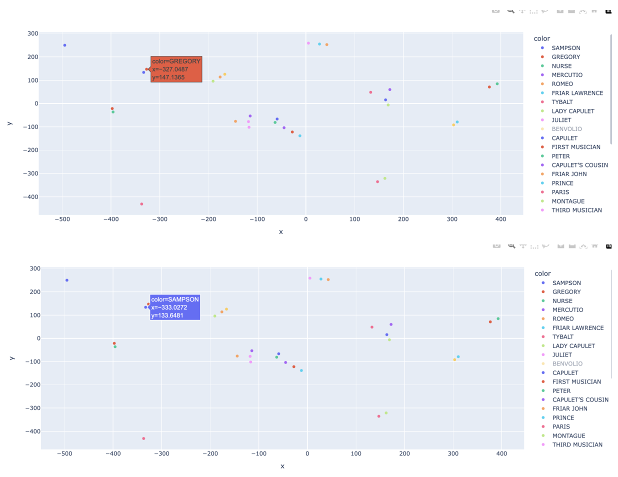


Figure 7: Gregory and Sampson's Embeddings

Embeddings of Character that Never Interact

The final embedding comparison we note is between Lady Capulet and Lady Montague. We highlight that these character never interact in the play, and thus would not have any biases from the training set. Their embeddings solely depend on their relations to other characters. We then highlight that their embeddings are far apart, which would be expected from their rival families.

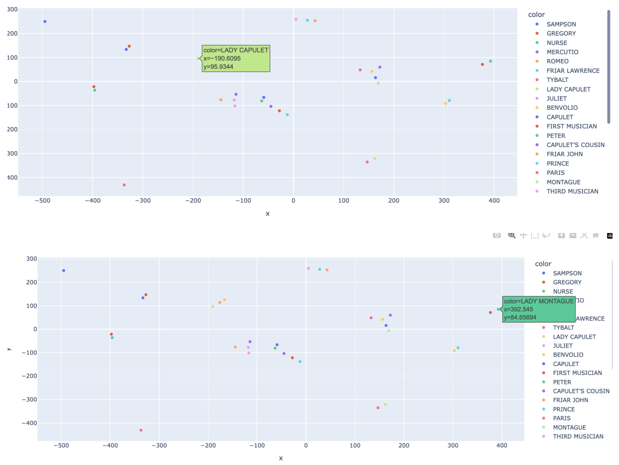


Figure 8: Lady Capulet and Lady Montague's Embeddings

Appendix B: Measuring Correctness through Test MSE

We also attempt to quantitatively measure our predictions. On our test set, we note that approximately 81% of data points 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.

The below graph shows percentage of test points within a distance from ground truth values

Appendix C: BardNet

The figure 11 shows the training loss on pretraining the BardNet architecture on Romeo and Juliet.

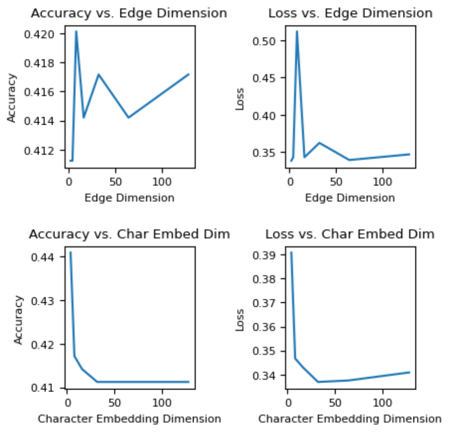


Figure 9: Hyperparameters against Loss and Accuracy

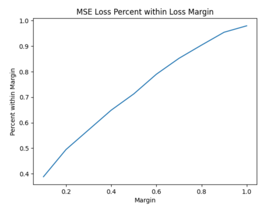


Figure 10: Percentage of test points within margin euclidean distance of ground truth

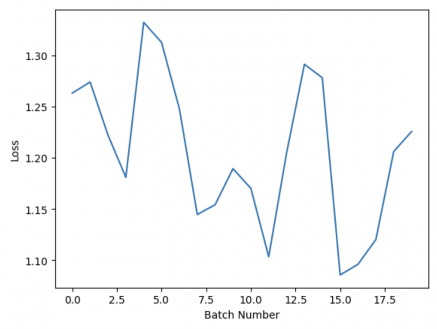


Figure 11: Training loss for pretraining task on BardNet

Appendix D: TransformerBard

The below figures show the pretraining and fine tuning losses for the transformer

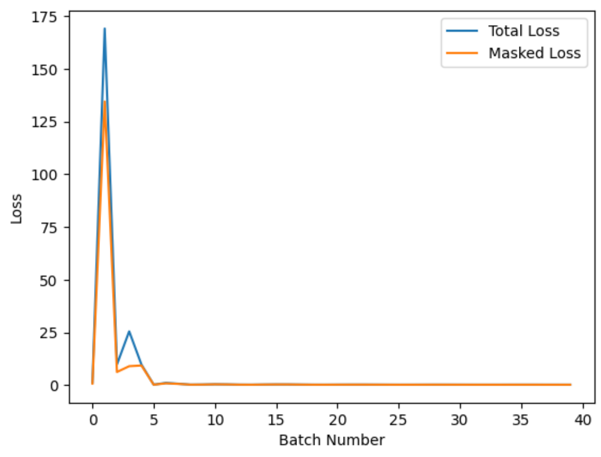


Figure 12: The masked and total training loss of our transformer. Masked loss is the loss on the masked dialogues

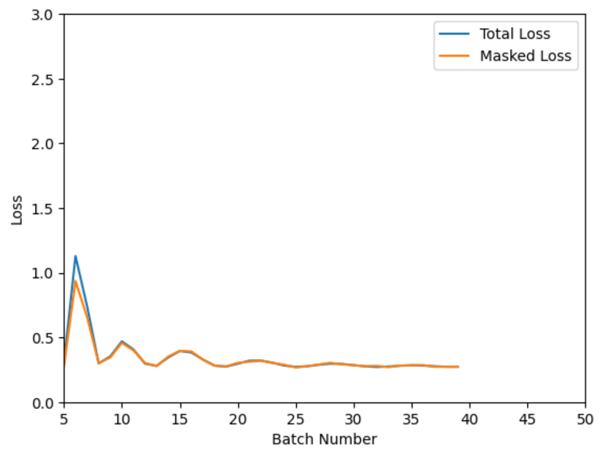


Figure 13: The masked and total training loss of our transformer in the tail of figure 5

Part IV

Team Contribution

Arushi implemented the BardNet Architecture and collected the Shakespeare data. She updated the architecture according to the reviewers' comments, and documented algorithms, evaluations, and hyperparameter tuning extensively in the final report. She conducted analyses on embedding closeness and clustering, sentiment prediction, and hyperparameter search.

Krishna implemented our transformer Architecture. He made several major experimental modifications to the BardNet Architecture to improve loss convergence and mitigate gradient explosions with the RNN implementation. He also generated visualizations and the corresponding analysis for the final report.

Christy conducted a literature review and identified articles that the initial architecture was inspired by. In a pair programming fashion, she debugged architecture issues with exploding gradients and designed the BERT-style pretraining transformer architecture. She helped compose the final report.

Andrew helped to extract relationships from Shakespearean characters, and developed the end-task of identifying familial relationships between characters. He implemented the Transformer architecture extension. He also helped compose much of the final report.

Part V

Supplementary Materials

`https://github.com/somaniarushi/ficembed`