Using Text-Level GNNs to Classify Explanations

Joe Shymanski

December 14, 2023

1 Introduction

The field of eXplainable Artificial Intelligence (XAI) is exploding with new research as transparency becomes more and more necessary for the future of AI. Thus, the field is always searching to utilize different methods to achieve this goal. In this paper, I detail one solution to the XAI problem I am researching which utilizes both network theory practices and machine learning tasks.

2 Background

I am conducting research which investigates the flaws of current techniques for evaluating XAI, specifically through the use of user surveys. In it, we have devised two classifications of explanations: actionable and placebic. In short, an actionable explanation is one which provides new, useful information to the user that was not previously known or obvious, while a placebic explanation is one which simply restates knowledge of which the user should already be aware.

I am curious to see if a model can accurately classify these two groups of explanations given only the text itself. The reason for this is because XAI models may output a range of explanations, and being able to label the output as either actionable or placebic can be immensely valuable to evaluating the model and later improving it, without the need of user studies. Thus, I began researching text classification techniques.

I stumbled across a number of methods which utilized GNNs in order to make these classifications. In it, they would represent the data as graphs in a number of different ways. All the graphs held in common that the tokens of the text were represented as nodes and the relationship between the words (adjacency, similarity, etc.) would be represented by the edges.

3 Method

The specific GNN model I am interested in testing is specified by Huang et al. [2]. In it, all the unique words in the dataset are added to the overall graph as nodes. Then edges are added between two words according to their proximity p to each other. The resulting "text graph" looks something like the one in Figure 1. In it, the set of vertices or the vocabulary of the dataset is given by V, and the number of dimensions in the word embeddings is represented by d.

The message passing mechanism proposed in the paper seeks to update a number of variables. The first is each node representation, which is done by taking a fraction of the current representation values and adding the remaining fraction of the message of that node. The message is calculated as the maximum values in each dimension of the edge weight and the neighbor node representation. The second is the aforementioned fraction, known as the information rate. The third is the edge weights themselves.

I was able to find a mostly-complete implementation of this model on GitHub [1]. However, there were some implementation details I had to vary in order for it to work with my dataset of explanations. I also downloaded several different sizes of GloVe embeddings, which is the word representation used by the code [3].

One noticeable difference that remained in all the experiments is the parallel between k

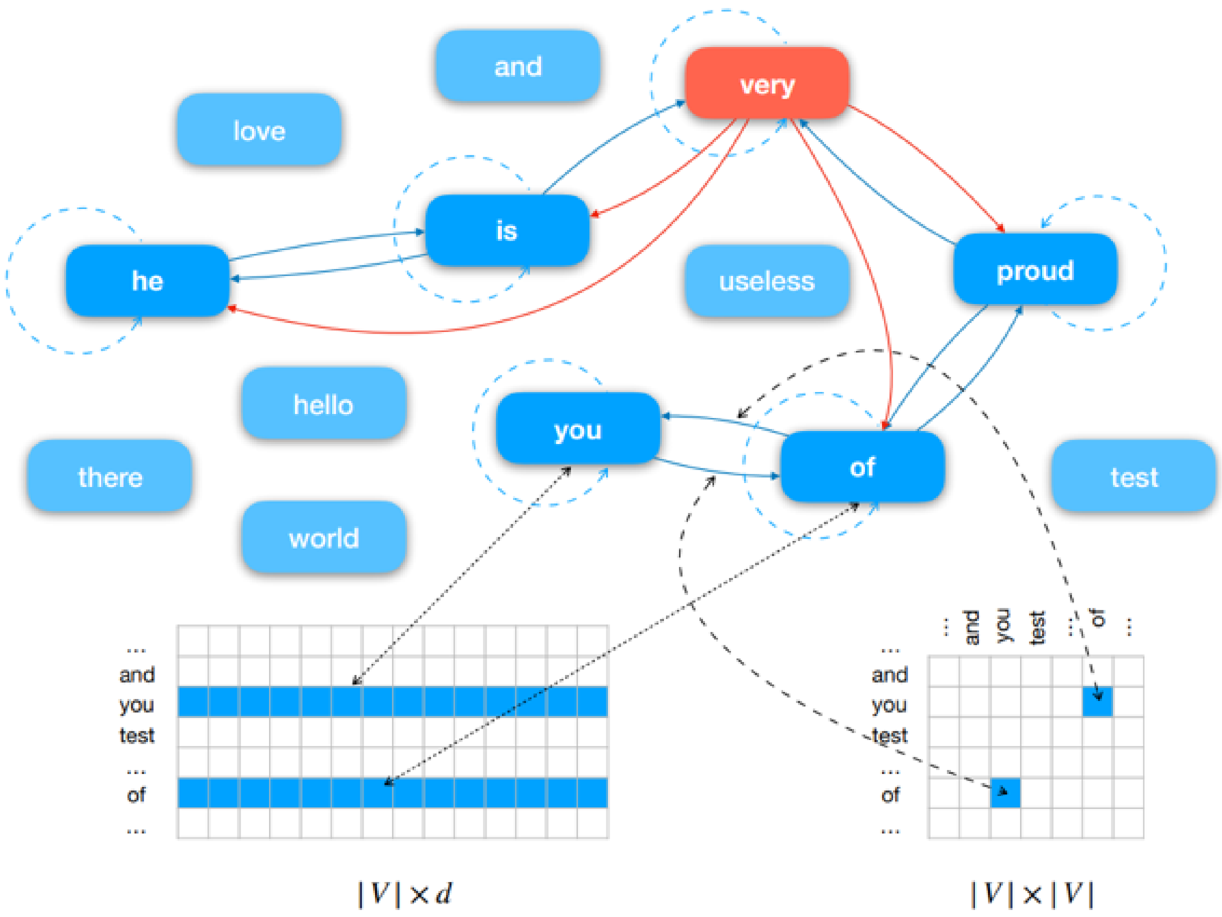


Figure 1: Example of a text graph.

parameter	value
minimum frequency	2
batch size	32
optimizer	Adam
learning rate	.001
weight decay	.0001
maximum epochs	300
early stopping patience	10 epochs
early stopping criteria	validation loss

Table 1: Model parameters and corresponding values.

in the paper and `min_freq` in the code. The former states that all edges which occur less than k times are included in a singular "public" edge, whereas the code filters all nodes which occur less than `min_freq` times into a singular public node.

There are a number of parameters for the model. Table 1 shows the parameters which were held constant throughout the experiment. Minimum frequency (`min_freq`) refers to the aforementioned minimum number of occurrences for a word to be included in the vocabulary, otherwise it is lumped into the public "unknown" token in order to properly train all parameters. The early stopping parameters refer to the method in which training will end before the specified maximum number of epochs if there has been no progress.

4 Dataset

The previous explanations that I have looked at using a user study is in the domain of chess. The goal is for the computer to send explanations to the human in order to explain the correct moves to a specified subset of chess puzzles. These puzzles all contain two moves which are both unambiguously the "correct" answer, meaning all chess models and experts agree that the moves are the most advantageous given the position.

An issue with this research endeavor is that there is not a large corpus of data from which to train the model. Instead, I had to handcraft 102 explanations and then clean each of them in order to format them the same way as the datasets in the paper. The resulting

# Train	# Validation	# Test	Labels	Avg. Length
58	14	30	2	9.74

Table 2: Summary of the dataset.

explanations averaged 9.74 tokens per explanation. The two labels (actionable and placebic) were evenly split. The training and testing sets were given a roughly 70/30 split, and then the validation set contained 20% of the remaining training cases. These values are specified in Table 2.

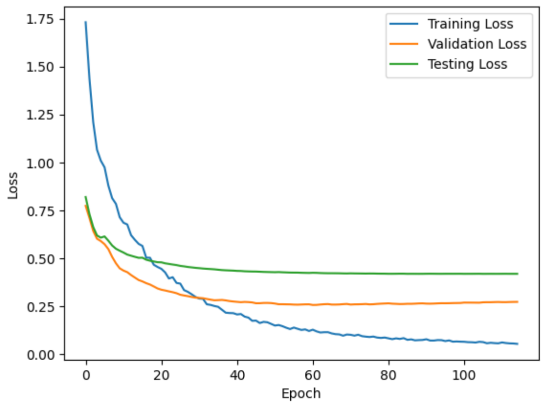
Because the size of this dataset is incredibly small, the results may not be as generalizable as I would like. However, I believe they still provide a small-to-medium-sized effect.

5 Results

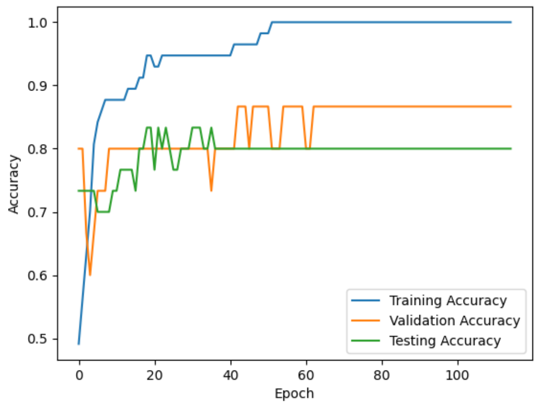
5.1 Best Embedding Size

Figure 2 shows the effect that the number of word embedding dimensions has on performance. Unsurprisingly, it appears that a larger embedding size leads to better loss and accuracy. Another interesting observation occurs in Figure 2c. Here, the loss seems to level off very quickly then suddenly decreases around 50 epochs and levels off again. This is the reason 100 warm-up epochs were specified for all tests, since this trend repeats itself often with the 300-dimensional word vectors.

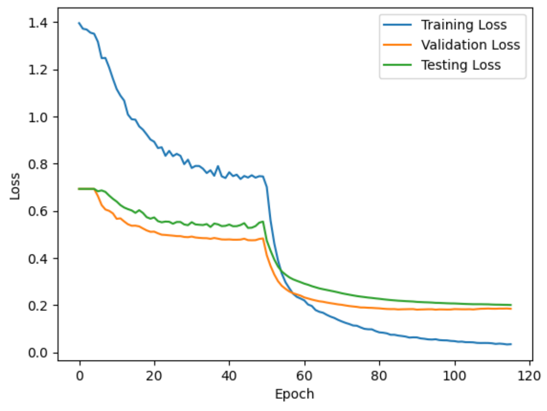
Another imperative observation is that the training time of the model does not increase significantly with the larger embedding size. The 300-dimensional embeddings increased training time by a fraction of a second, on average. In fact, each epoch only needed about .04 seconds to train between both embedding sizes. By far, the most time consuming task was loading the GloVe embeddings at the beginning of the training session, which depended on the number of tokens in the GloVe vocabulary (400,000).



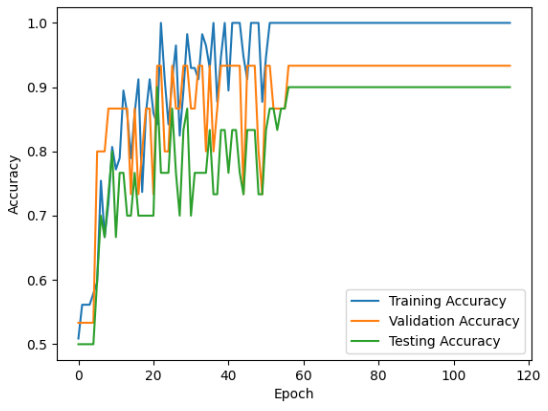
(a) Loss for 50-dimensional embedding.



(b) Accuracy for 50-dimensional embedding.



(c) Loss for 300-dimensional embedding.



(d) Accuracy for 300-dimensional embedding.

Figure 2: Performance for different embedding sizes.

Training (%)	Validation (%)	Testing (%)
99.7 ± 0.1	91.7 ± 4.9	88.0 ± 3.5

Table 3: Accuracy of the best performing model.

5.2 Best p Value

Figures 3 and 4 show the effect that p has on performance. As a reminder, p specified the number of preceding and succeeding neighbors to connect with an edge to the current word. The trend seems to indicate that a p value around 2 or 3 leads to the smallest loss values and largest accuracies in the validation and testing sets.

5.3 Best Performing Model

The best performing model utilized the 300-dimensional GloVe word embeddings and a p value of 3. 100 warm-up epochs were enforced. Table 3 shows the model's performance on all three datasets.

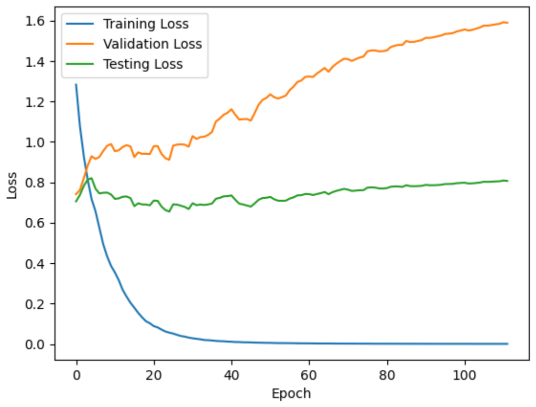
6 Discussion

The trends found in the analysis of the p values and word embedding sizes corroborates the findings of the original paper. Those authors showed that p values between 2 and 5 provided the best performance, and that the largest word embeddings did as well.

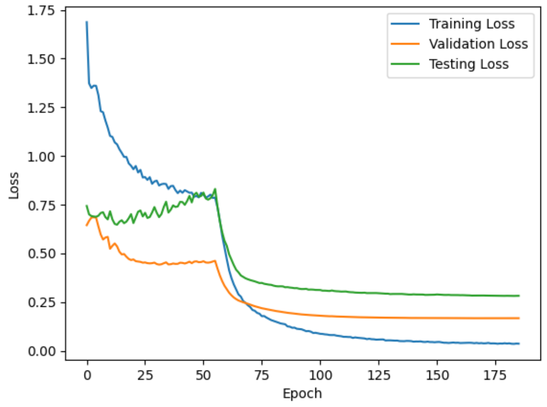
The final accuracy measures give significant hope in the efficacy of the model. The paper was able to achieve roughly 98% accuracy on a larger dataset with 8 labels, so I am confident a larger dataset would boost the testing accuracy closer to this number.

7 Conclusion

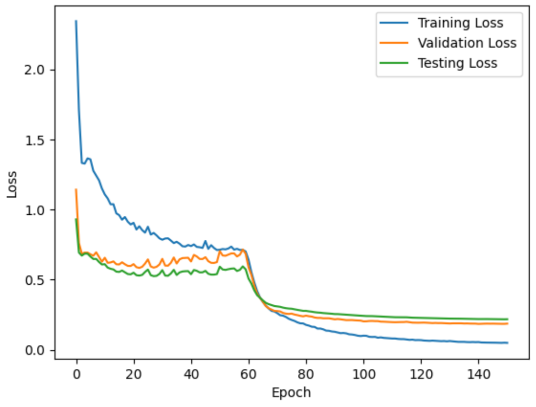
In conclusion, I believe this model is actually quite effective at classifying the explanations created by an XAI model as either actionable or placebic.



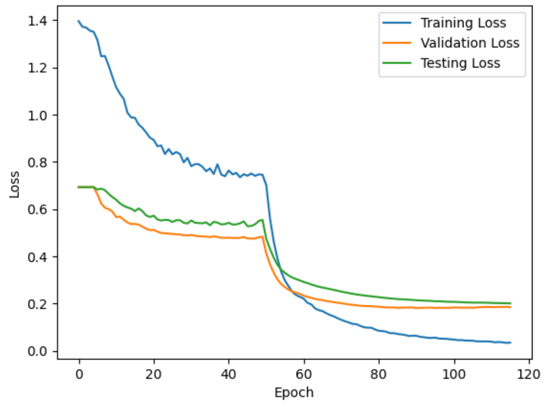
(a) $p = 0$.



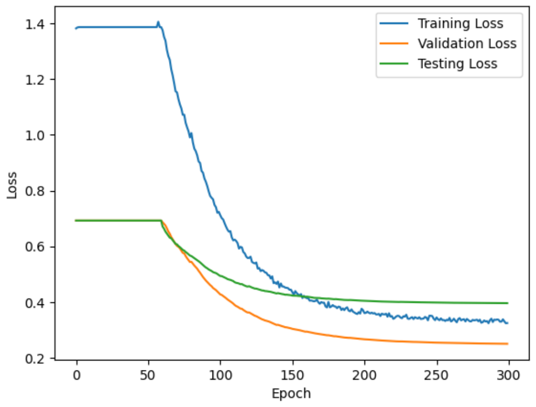
(b) $p = 1$.



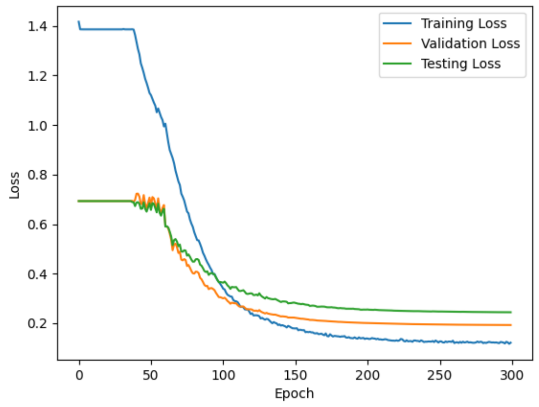
(c) $p = 2$.



(d) $p = 3$.

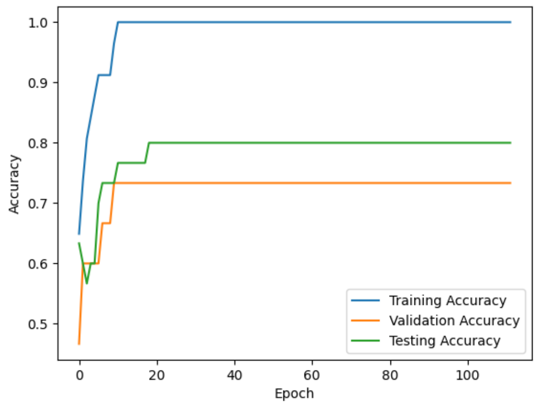


(e) $p = 4$.

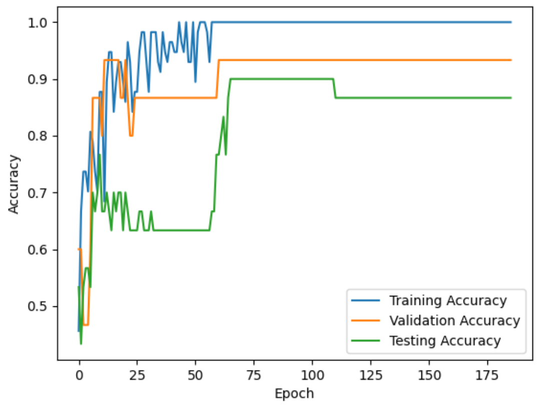


(f) $p = 5$.

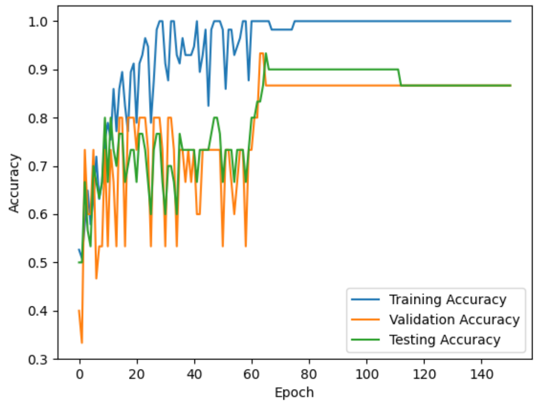
Figure 3: Loss for different p values.



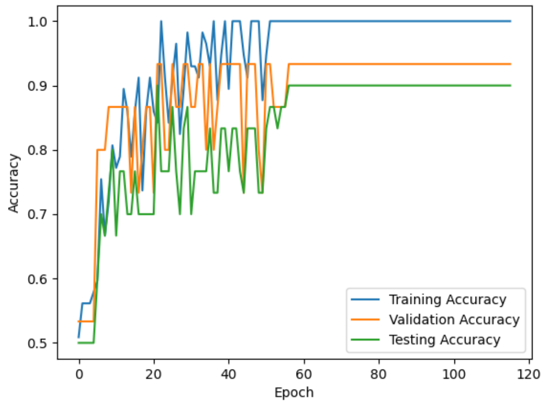
(a) $p = 0$.



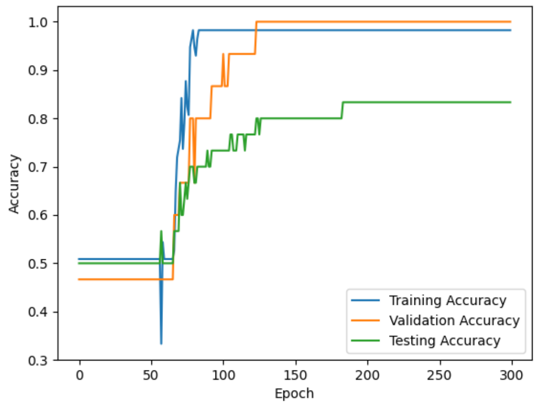
(b) $p = 1$.



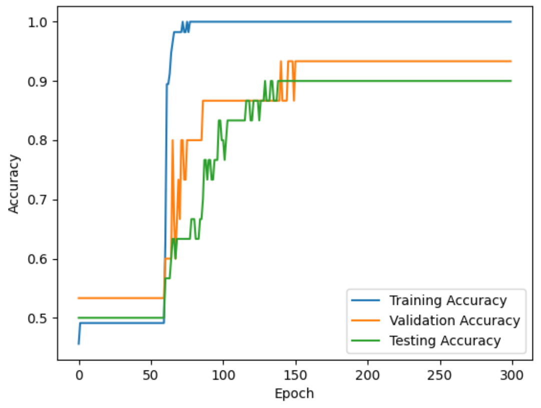
(c) $p = 2$.



(d) $p = 3$.



(e) $p = 4$.



(f) $p = 5$.

Figure 4: Accuracy for different p values.

There is plenty of future work which could help support the findings of this study. First, enhancing and enlarging the dataset could amplify the results and the subsequent claims about the model. A labeled dataset of roughly 500-1,000 explanations could suffice. Second, generalizing the explanations included in the dataset would help ensure this model could work across many domains, not just the current chess domain.

References

[1] Cynwell. Cynwell/text-level-gnn: Text level graph neural network for text classification.

[2] Lianzhe Huang, Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. Text level graph neural network for text classification. *arXiv preprint arXiv:1910.02356*, 2019.

[3] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.