Predicting guilt judgments from crime stories

Elisa Kreiss
Department of Linguistics
Stanford University
ekreiss@stanford.edu

Judith Degen
Department of Linguistics
Stanford University
jdegen@stanford.edu

Christopher Potts
Department of Linguistics
Stanford University
cpotts@stanford.edu

Abstract

[ek: update abstract wrt new framing] News has become a central part of our lives and with it our assessments and reproductions of these stories. However, the way in which humans assess guilt on the basis of such crime stories is unclear. If neural networks can learn to predict guilt judgments from a news story, they can help us form hypotheses about relevant features for human guilt predictions. However, whether neural networks can even learn to predict guilt judgments from a given news story is still an open question. In this paper, we will present a bidirectional LSTM with self-attention that makes a promising first step to predict guilt judgments. On the basis of attention visualizations, we argue that the model learns meaningful patterns from the data that guides its prediction.

1 Introduction

[ek: update according to framing proposal in email]

Deep learning models are increasingly applied to language tasks not only to develop technologies but also to derive new insights about language.

However, deriving scientific insights presupposes that the networks are learning sensible representations, which is often in question. Attention mechanisms have improved model performance and also made our networks more interpretable.

In this paper, we apply neural networks with attention to a task that has both linguistic and societal import: predicting whether the reader of a narrative text about a crime will conclude that its main subject is guilty or innocent.

Our models are good at this task, which is nice. However, we are, in particular, interested in seeing how our networks attend to markers of certainty and uncertainty in these texts. To what extent do these epistemic markers guide network decisions?

1.1 Related work

The central challenge is to find a promising neural network architecture that can extract relevant features of an input paragraph. This can then be used to predict a value that represents a guilt judgment.

Word embeddings There are a variety of systems that investigate how to find meaningful word representations. In the literature, GloVe word embeddings (Pennington et al., 2014) have been widely used to represent word meanings. Those embeddings are context-independent, i.e., each word only has one respective vector representation independent of the context it occurs in.

More recent advances in word embeddings aim to provide context-sensitive word representations to obtain richer and more informative word meaning representations (e.g., ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018)). The BERT word embedding model is a Transformer based network which incorporates the context before and after the critical word to obtain a word representation. Devlin et al. report that using these embeddings boosts performance in all of the 11 natural language tasks it was trained on compared to previous systems.

Paragraph embeddings While word embeddings are useful, we need to arrive at a meaningful **paragraph** representation for our purposes. In other words, the word embeddings need to be combined in a meaningful way to arrive at a more complex representation.

For short phrases, bidirectional LSTMs (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997; Graves and Schmidhuber, 2005) have been shown to be a useful tool. However, they have difficulty resolving long-term dependencies, which is why their performance decreases with longer input paragraphs (Hochreiter et al., 2001). (Vaswani et al., 2017) developed am attention mechanism to resolve these issues in Transformers. However, a similar principle can be applied to LSTM architectures as well.

(Lin et al., 2017) introduce a bidirectional LSTM with self-attention that receives word embeddings as an input and outputs a paragraph representation. The bidirectional LSTM receives the GloVe embeddings of the paragraph as input. The self-attention layer is the weighted sum of all of the LSTM's hidden states. But instead of using a single weighted sum to arrive at a vector, they suggest to perform this multiple times with different weight vectors to arrive at an attention matrix. Furthermore, they introduce a penalization term which ensures that each attention vector in this attention matrix focuses on different semantic features of the input paragraph. Which semantic features are more important than others can in turn be learned again, dependent on the task it needs to solve. Lin et al. report that the model outperforms previous models in author profiling, sentiment classification and textual entailment.

2 Dataset

For training and testing the model, we use the Annotated Iterated Narration Corpus (AINC) collected by (Kreiss et al., 2019). The corpus was created for investigations on how news stories change when they are propagated from one person to another. First the authors created five stories of approximately 850 words, which are called the *seed* stories. All of them report a crime and an arrest of one or more suspects. Each of these five seed stories exists in a weak and a strong evidence condition. The stories are identical up to the last phrase which then either raises doubts about the arrest or emphasizes its validity. For example, if a suspect was arrested on the basis of camera footage, this footage was described as being of "very poor quality" in the weak evidence condition and "very high quality" in the strong evidence condition. Each story was given to a participant who was asked to read and afterwards reproduce it. The reproduced story was then given to the next participant who again reproduced it. This pattern was repeated for 5 generations of participants. The data collection therefore followed the transmission chain paradigm, introduced by (Bartlett, 1932). Following this schema, each story and condition was reproduced in 5 chains over 5 generations, resulting in 250 reproductions, and therefore 260 stories overall (see Figure 1).

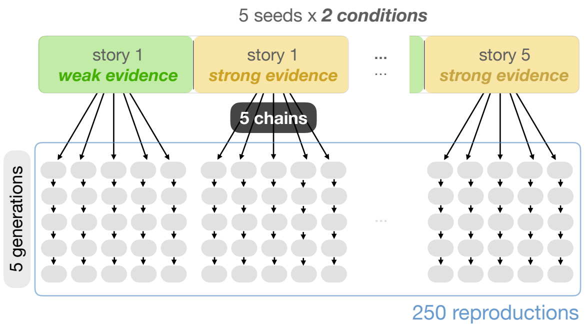

Figure 1: Corpus collection schema as presented in (Kreiss et al., 2019).

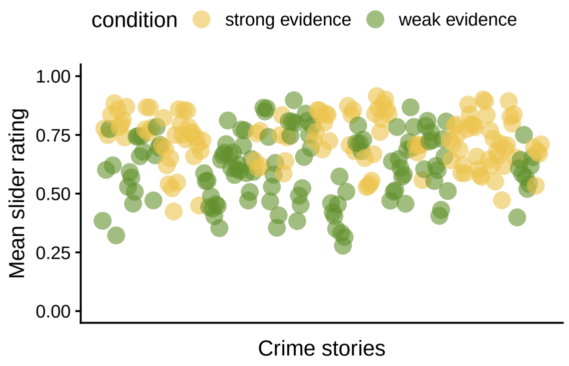

Figure 2: A point represents the mean subject guilt rating for a story in the corpus. They are color-coded with respect to their condition.

After the corpus collection, Kreiss et al. annotated the corpus with human judgments. The questions were primarily related to different aspects of guilt perception but also for example perceived subjectivity of the story writing. Participants were recruited on Amazon Mechanical Turk and indicated their response on a continuous slider, here underlyingly coded as ranging from 0 to 1. Most interestingly for this project, they asked "How likely is it that the suspect is / the suspects in the crime are guilty?". Each story received approximately 20 ratings, ranging from 0 to 1. For the purpose of this work, we will consider the mean rating for each story as its guilt judgment label. The labels of each story are shown in Figure 2. In contrast to the raw ratings, the means only range from 0.27 to 0.92.

In summary, the AINC is a corpus of reproduced news stories, annotated by human guilt judgments. Since the stories originated from only 5 unique stories (each in 2 conditions), a lot of information is shared between single data points. Despite this similarity, the range of guilt judgments is still high, alluding to subtle differences that trigger this variance. One of the challenges, this dataset poses, is its size. There are only 260 labeled examples that can be used for training and testing a model. Can a bidirectional LSTM with attention pick up on the patterns in human judgments with only very limited available data?

3 Model

The goal of this work was to determine whether neural networks can be a useful tool to investigate the formation of guilt judgments. To do this, we used a model which was based on the design proposed by Lin et al.. We chose to use this model as our base because of its promising performances in various NLP tasks. Furthermore, its attention module provides a straightforward way to investigate the inner workings of the model.

We made two major adjustments to the model proposed by Lin et al.. Firstly, we simplified the attention module to represent a vector and not a matrix. Second of all, we replaced the GloVe word embeddings by BERT embeddings[1].

The following sections will explain the model architecture and the training of the model in detail.

3.1 Architecture

The tested model (see Figure 3) receives a crime story as an input and outputs a value between 0 and 1. This value is interpreted as a continuous guilt assessment of the suspect in the story.

Overall, the model consists of four main modules (see Figure 3). First the crime story is transformed into **word embeddings**. These are the input to a bidirectional **LSTM** with **attention**, which will output a single vector representation for the whole input sequence. To obtain the desired output in the range [0, 1], a simple **logistic regression** is performed on the vector.

Word embeddings (BERT) We used a pretrained BERT model as the word embedding module (represented by the yellow box in Figure 3). The module receives the original crime story as an input and returns a word embedding for each

[1]In their paper, Devlin et al. showed that BERT embeddings (can) improve existing NLP systems. Pilot runs in the development of the model showed that BERT improved the convergence of the model.

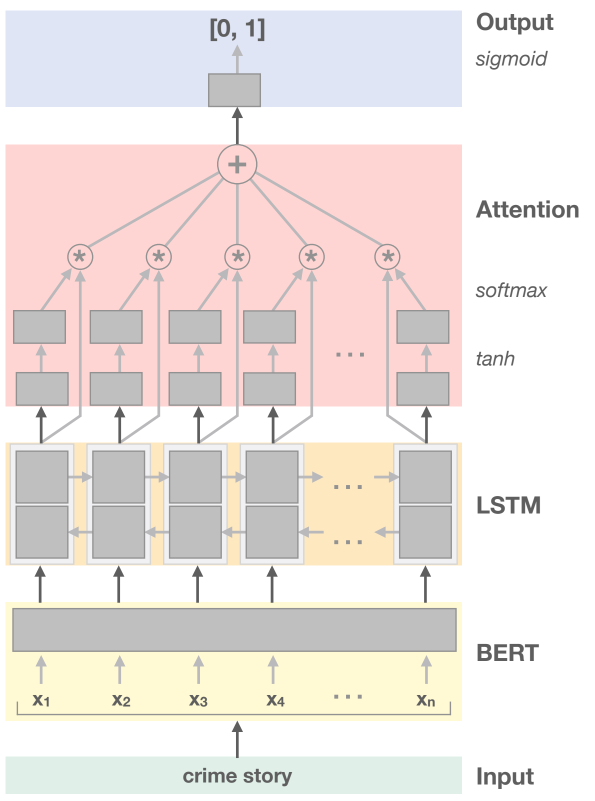

Figure 3: Proposed model architecture to predict guilt judgments (between 0 and 1) from news stories. The arrows indicate the information flow. The boxes stand for operations performed on the data.

token in the story $(x_1, x_2, ..., x_n)$. For tokenization, we used the BertTokenizer implementation in the python package *pytorch-pretrained-bert* [2]. The BERT word embeddings for each token were extracted from the Google-pretrained base BERT word embeddings (uncased) with 768 hidden layers. Note that given the requirements of the pretrained BERT model, the original story had been transformed to all lower-case letters.

This module takes the raw crime story as input and returns a word embedding representation for each of the n tokens in the text. The output is therefore of size $n \times 768$.

LSTM The bidirectional LSTM (represented by the orange box in Figure 3) takes the word embeddings as input to enhance the contextual representation of the story.

The LSTM has 200 hidden dimensions. Since a bidirectional LSTM concatenates the hidden states of the forward and backward LSTM, each hidden layer has 400 hidden dimensions. Therefore, the LSTM module receives an input of size $n \times 768$ and returns a matrix H of size $n \times 400$.

Attention The attention module (represented by the red box in Figure 3) creates a weight for each input token before they are summed over. The boxes inside the attention module stand for linear operations that reduce the dimensionality of the input. Tanh and softmax are non-linear transformations on the output of these dense layers.

The LSTM module returns H, i.e., n hidden states of size 400. These are first linearly reduced to $n \times 50$ dimensions and tanh transformed.

$A_1 = tanh(W_{a1}H)$[ek: Lin has here H^T, but this is not true for our model]
The resulting matrix A_1 is further compressed to the attention weight vector $\mathbf{a_w}$ of size $n \times 1$. The softmax ensures that its values sum to 1.

$\mathbf{a_w} = softmax(\mathbf{w_{aw}} A_1)$

Now we take the dot product of the LSTM hidden state matrix H and the just obtained vector a_w to obtain the attention output vector $\mathbf{a_{out}}$ of size 400.

$\mathbf{a_{out}} = a_w^T H$
This is the output of the attention module.

Output (Logistic regression) The logistic regression module is represented by a blue box in Figure 3. For a prediction, the output of the attention module ($\mathbf{a_{out}}$) is linearly transformed into a single number. The Sigmoid function ensures that the prediction lies between 0 and 1, just like the restriction on the guilt judgments it is trained on.

Overall, the model has 111,054,742 trainable parameters. Only 0.02% of them are in the attention and output modules, and 1.40% are in the LSTM module. These parameters were initialized randomly. The rest of the parameters (98.59%) are the pretrained weights in the BERT word embedding module.

In summary, the model receives a crime story as an input. BERT word embeddings for the story feed into a bidirectional LSTM. The attention module computes the weighted sum over the LSTM's hidden states. The resulting attention vector is fed through a linear layer and a Sigmoid function to ensure that the prediction lies between 0 and 1.

3.2 Training

To begin with, we randomly held back 26 stories (from the 260 stories in total) as the final test set. This helps to assess the model performance on a set of stories which has never been evaluated in the process of model development. The remaining 234 stories were then for used for model training and validation, which was done using 10-fold cross validation. The cross-validation results inform us about the model variation given various training/validation splits. For example, it is quite likely that given the small size of the dataset, this variation could be quite high. Note that, each fold had either 23 or 24 stories in it, each of which was associated with a guilt rating that functioned as the target label. Recall that this guilt rating is the mean subject guilt rating obtained from the previously described data collection and annotation.

For training, the model uses mean squared error (MSE) as the loss function, stochastic gradient descent as the optimizer and a learning rate of 0.1. The whole model was implemented using pytorch (Paszke et al., 2017). It was trained for 30 epochs, for each of the 10 training/dev-test configurations in the cross-validation.

4 Experiment

This work investigates whether a bidirectional LSTM with self-attention (as described in Section 3.1) can predict human guilt judgments from

[2]Huggingface Inc., https://github.com/huggingface/pytorch-pretrained-BERT, June 09 2019

news stories. In each step of the 10-fold cross-validation, the model was trained on 206/207 stories and 23/24 were held out for dev-testing. Each story has an associated suspect guilt rating, which is the mean rating obtained from human judgments. This value functions as the target label.

Since the target labels only range between 0.27 and 0.92, one potential problem might be that the model only learns to predict the overall mean of the labels in the training data. We would also expect this outcome if the model does not find any features that it can take as predictors for guilt ratings. This is why, the baseline model to test the dev-test data on will simply predict the mean of the training data labels. If the our model outperforms this baseline, we will consider it worthy of further investigations.

Both model performances will be evaluated on the mean-squared error (MSE) of their prediction to the target label on the cross-validation test set.

4.1 Results

Figure 4 shows the mean squared error (MSE) loss for each training epoch. Firstly, the training loss (in blue) is approaching a loss of 0 toward the end of the training in all cross-validation configurations. Thus the amount of training epochs seems to be sufficient. Crucially, the dev-testing loss (in orange) is overall lower after the last training epoch than the baseline (in yellow). Therefore, the model obtains information from the dataset that it can use to improve its predictions. However, there is also a high amount of variation between the different cross-validation steps. The mean of the training labels alone (i.e., the baseline) only has a very small loss on cross-validation configuration 2 and the trained model can barely beat it. This is in clear contrast to cross-validation step 4, where the baseline model loss is very high on the dev-test data and the trained model can easily surpass it. Those two cases exemplify the high variation that comes with the different splits of training and dev-testing data[3].

But looking at the MSE loss alone is not on its own sufficiently informative to assess how well the model actually learns to predict the underlying distribution. Figure 5 shows the correlation between the actual target label (on the x axis) and the model prediction (on the y axis) for cross-validation step 0. Before training, the training and dev-testing data has a high variance around the perfect correlation line ($r = 1$). The model only predicts values between 0.48 and 0.79. However, the underlying labels range from 0.31 to 0.92. Even though the Pearson correlation still predicts a rather high correlation on the testing set ($r = 0.65$), the mean squared error is comparably high with 0.022.

Already qualitatively, the model predictions after training (Figure 5) seem far more informative. Now the model predictions have approximately the same range as the target labels (between 0.33 and 0.91). The Pearson correlation itself is now 0.85, and the mean squared error reduced to 0.007.

Since the baseline model does not learn over training epochs, its predictions stay the same.

Qualitatively, these plots appear very similar throughout all cross-validation configurations and can be compared in Figure 9 and 10 in the Appendix. However, quantitatively the correlation on the dev-test set changes. Those differences are mainly driven by outliers.

Overall, the mean correlation between the model prediction and the human judgment on the testing set across all cross-validation steps is 0.68. When we collapse over all cross-validation folds and examine the loss after training, the difference in loss between the model predictions and baseline is significant ($p < 0.0001$).

As a final performance evaluation, we can examine the target-prediction correlation on the held-out test set (see Figure 6). For predictions on the test set, we used the final model weights obtained by the first cross-validation configuration. This setup was chosen, because of the low dev-testing loss and the high correlation of the dev-test set with the target labels[4].

The Pearson correlation on the held-out test set is still 0.84 and almost identical with the performance on the dev-test set. This fairly high correlation and the fact that the high correlation reproduces with the held-out testing data indicates that the model learns something meaningful that can generalize well.

5 Model analysis

We have seen that the proposed model can predict human guilt judgments when given a crime

[3]However, I cannot exclude that also the random parameter initialization plays a relevant role here. But at least the differences in the baseline are definitely caused by the different splits.

[4]This was the only evaluation that was performed on this held-out test set.

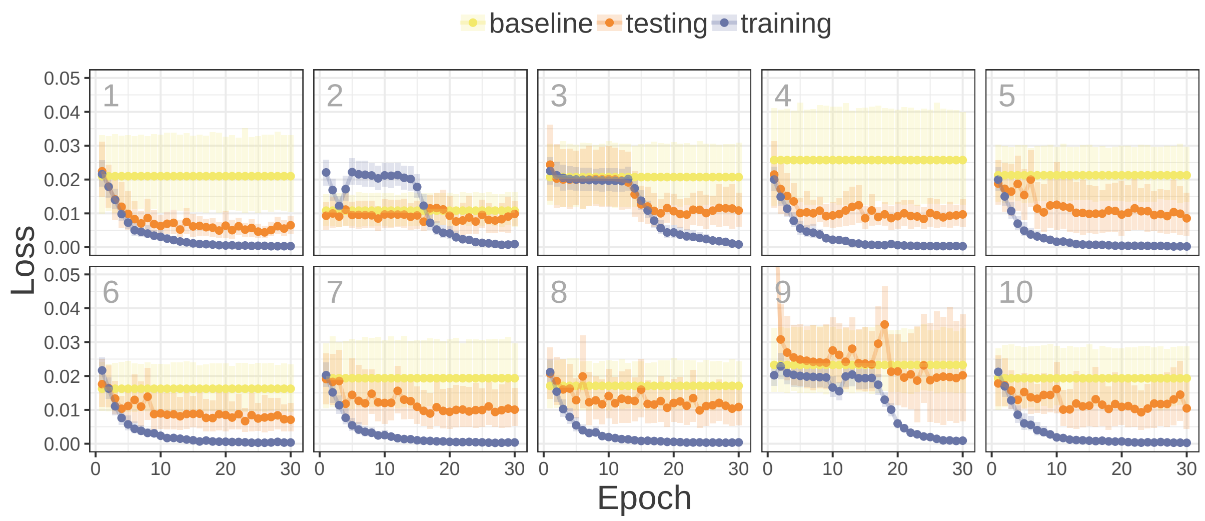

Figure 4: Loss (mean squared error) over epochs (x axis), faceted over cross-validation configurations. The performance of the model on the training set (in blue) approaches zero. The performance of the model on the dev-test set (in orange) generally outperforms the baseline (in yellow).

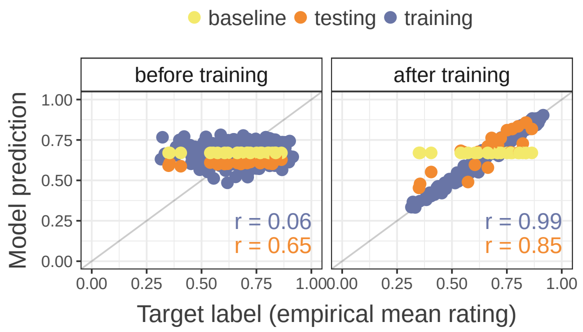

Figure 5: Correlation between target label (x axis) vs. model prediction (y axis) before and after training.

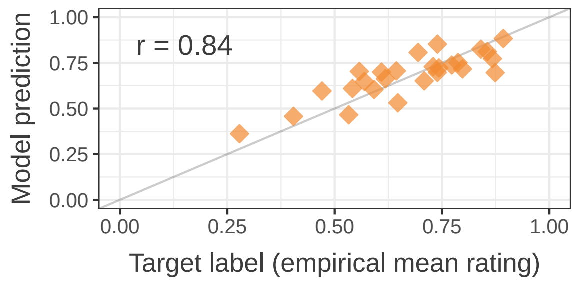

Figure 6: Testing label (x axis) vs. model prediction (y axis) after training on the held out test set (26 data points) using the parameter settings obtained after the 30th epoch from the first cross-validation. Pearson correlation of 84%.

story. However the question remains whether there are patterns that underlie these predictions and whether we can interpret them. This is especially necessary, if we want the model to inform hypotheses about the processes that possibly underlie human judgments.

5.1 Visualization

First, we need to investigate whether the model seems to pick up on patterns that are meaningful and possibly even interpretable for us.

In the original introduction of the attention mechanism, Vaswani et al. suggested that a visualization of the attention weights can inform us about what affects the network's prediction. We follow this approach to look for first indications of a meaningful representation in the $\mathbf{w_{aw}}$ attention weight vector described in Section 3.1. Since the softmax forces the sum of the weights to be 1, we cannot interpret the weights on their own for each word or across stories. Instead, the relevance lies in the differences between words and phrases within each story and pattern similarities between stories.

To investigate what might affect model predictions, we ran the model again on the final test data (as described in Section 4.1). Figure 7 displays three of these stories with their attention weight distribution.

The model seems to focus on phrases that explicitly describe uncertainty about the evidence (see Figure 7C). If present, they usually outweigh

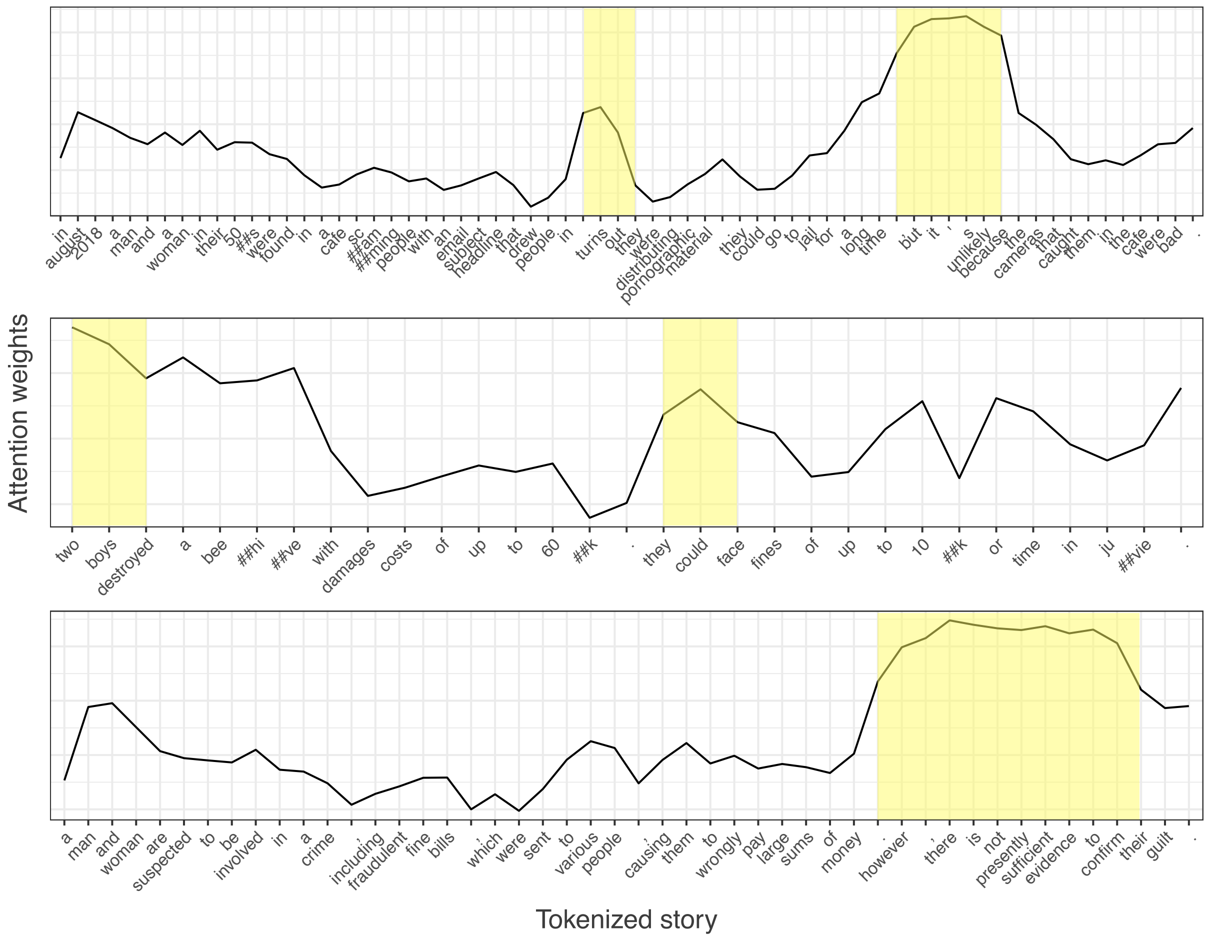

Figure 7: Visualization of the attention weights (y axis) for a tokenized story (x axis) from the test set. Because we took the softmax over the attention weights, the scale of the y axis is irrelevant. Interesting aspects are marked in yellow.

the rest of the story. This suggests that if there is an explicit claim that affects the evidence of the suspect's guilt, it is considered as the most important source to inform guilt judgment.

Additionally, peaks occur on words and phrases which communicate turning points in a story, such as "however", "but", "even though" and "it turns out". This can be seen in all three stories in Figure 7. Those phrases could be relevant because they usually indicate a contrast to a prior narrative. Since those markers mostly follow reports of arrests, they might be strongly correlated with objections to those arrests which would in turn influence guilt perceptions.

Figure 7B shows a case where the model seems to find a simple declarative ("two boys destroyed") to be relevant for the final prediction. This is especially interesting, since declaratives on their own do not generally communicate guilt-related information. However, they are very important for guilt judgments because they do not allow any uncertainty about the association between crime and suspect.

In summary, the visualization of the attention weights contribute further evidence that the model learns meaningful patterns in the data.

5.2 Qualitative analysis

The model visualization provides evidence for the claim that the model picks up on meaningful patterns in the corpus. This allows us to use the model to inform new hypotheses.

But does the model make reasonable predictions on newly constructed data points? The original corpus started out with five different crime stories. Each of these stories occurred in two conditions – one suggesting that the evidence that led to the arrest was weak, and the other that the evidence provided a strong case. Additionally, the stories were filled with uncertainty markers such as "allegedly" or "(un)likely". The corpus cannot inform us about the relationship between those uncertainty markers and the evidential manipulation. What happens if we ask the model to predict a guilt judgment for each of the original stories without those uncertainty markers/hedges?

To investigate how hedging influences model predictions, we rewrote those 10 original stories into versions without any uncertainty markers. Note that they remained as close to the original as possible, while still remaining grammatical.

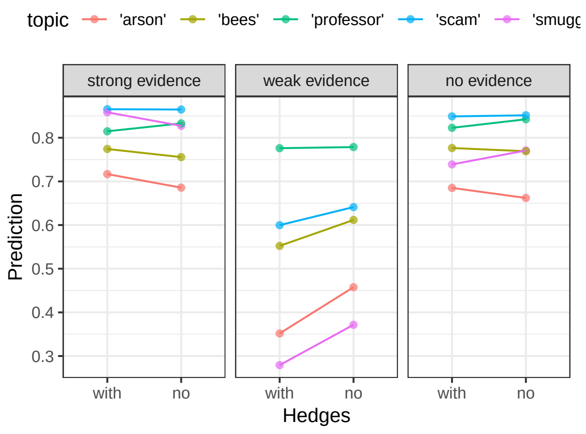

Figure 8: Model predictions on the relationship between uncertainty markers and evidence statements. [ek: fix caption and make figure more readable]

Figure 8 shows the results of this analysis. The results suggest that uncertainty markers have different effects on guilt prediction in the two evidence conditions. When the evidence is strong, removing all uncertainty markers does not affect the guilt judgments. However when the evidence is weak, removing those hedges increases the guilt judgment.

This allows for an intuitive interpretation. When evidence already overwhelmingly speaks for the suspect's guilt, it outweighs the hedges. However if the evidence is questionable, other sources of uncertainty are considered to inform a final judgment.

Notably if the evidence manipulation is excluded, the model predicts guilt judgments in the range of the strong evidence condition. In other words, the model predicts that a reassurance that the arrest is justified does not change the guilt judgments. However, formulating explicitly that there are reasons to question the arrest, causes the guilt assessments to drop. When we remove the hedges again, we cannot see a common structure to the change in ratings (possibly more similar to the strong evidence ratings though).

5.3 Conclusion

In this section we showed that our attention module provides intuitive insight into the model workings. Furthermore, the model makes interesting predictions on new data.[ek: ...]

6 Discussion

The results show that when given a news story, a bidirectional LSTM with self-attention can already capture something essential about human guilt judgments. Especially considering the limited amount of data (only 236 data points in the training set), the model can already extract meaningful features for guilt judgment predictions.

So far, the model has not been subjected to hyperparameter tuning. It is likely that this will further boost the model's performance. Furthermore, it is plausible that training and testing the model on the guilt ratings directly instead of their means can further increase the performance, since only showing the means hides a vast amount of information about the underlying distribution of ratings which can be very meaningful. For example, a news story where participants highly vary in their judgments should influence the model predictions less than a story with low variance.

The introduced model is a promising first step to develop a better understanding of human guilt judgments. I am confident that in the future models like these can be used to inform hypotheses about what humans consider for their assessment. This can in turn inform us about how different reporting on a news story will influence these assessments.

Acknowledgments

I'm particularly thankful for the valuable feedback from Sebastian Schuster. [ek: any grants to cite here, Judith?]

References

Frederic C Bartlett. 1932. Remembering: An experimental and social study. *Cambridge: Cambridge University*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Elisa Kreiss, Michael Franke, and Judith Degen. 2019. Uncertain evidence statements and guilt perception in iterative reproductions of crime stories. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 41.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

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

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

7 Appendices

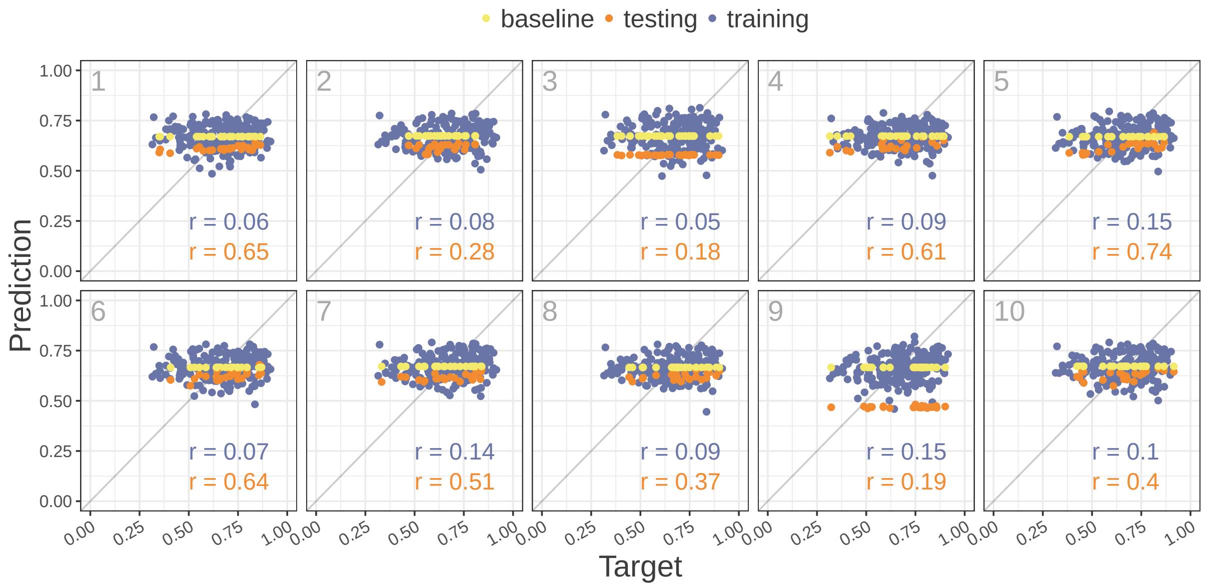

Figure 9: Testing label (x axis) vs. model prediction (y axis) before training; faceted over cross-validation configurations.

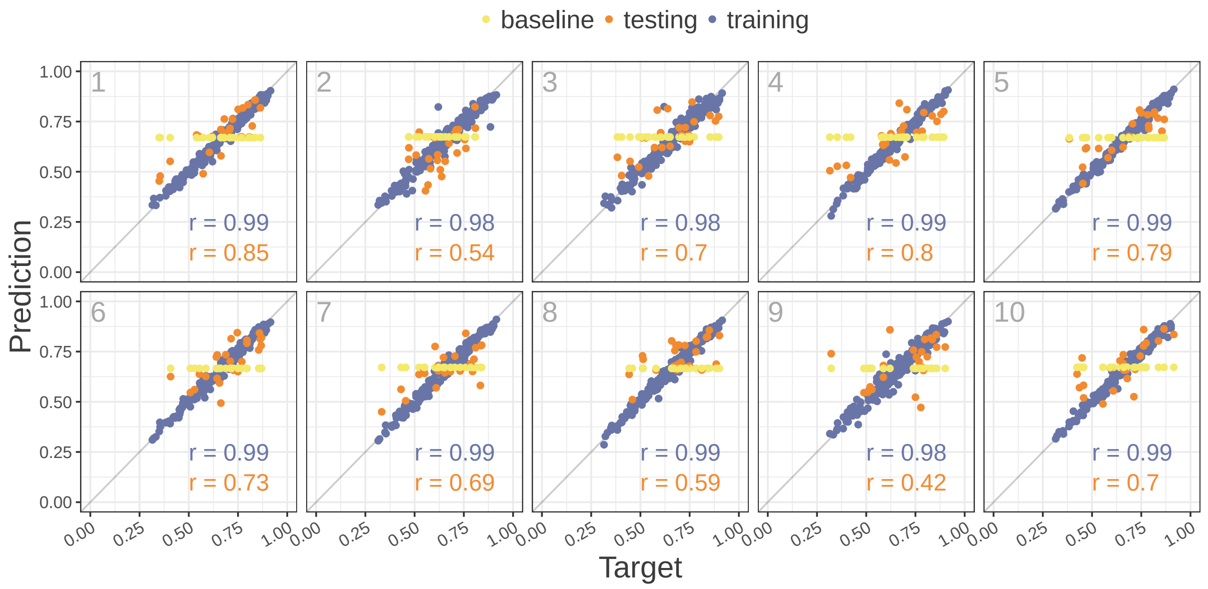

Figure 10: Testing label (x axis) vs. model prediction (y axis) after training; faceted over cross-validation configurations.