A AMT Annotation Notes

We ran AMT tasks in batches. We performed quality control tests after every 1-2 batches and banned workers who were consistently performing poorly (and removed their responses from our dataset). For quality control, we used a combination of automatic methods (SGD over interannotator agreement using the quality control objectives described in the supplementary material of Yatskar et al., 2016) and manual methods (e.g. searching for workers who always selected the same answers, who had strange answering patterns, who copied-and-pasted free responses) to identify workers with poor-quality work. Work by such annotators was reviewed, and these workers were banned if they were repeatedly not following instructions on a number of HIT's.

B Emotion Agreement Matrix

We include a NPMI confusion matrix for aggregated Plutchik paired responses in Figure 7. Black boxes signify the same emotion but at different intensities (high vs. moderate). In general higher co-occurring responses are along the diagonal. However, we note that there are two main clusters coinciding with strongly positive emotions (joy and trust) and strongly negative emotions (anger, disgust, and sadness) where disagreements are more likely to occur. To a lesser extent, there also is a slight co-occurrence between fear and the strongly negative emotions.

C Model Implementation Details

All classification models are trained with the Adam Optimizer (Kingma and Ba, 2015) with a learning rate 0.001 and gradient clipping if the norm of the gradients exceeds 1. We regularize with dropout layers whose probabilities are specific to each model. All models are trained with word embeddings of dimensionality 100 that are initialized with pretrained Glove vectors (Pennington et al., 2014). For classification labels, we use the majority label among annotators for a particular character-line pair.

C.1 LSTM Classifier

We train a 2-layer bidirectional LSTM encoder. The hidden states of the LSTM have dimensionality 100. We add dropout layers with p=0.5 in between the word embedding layer and the LSTM and between LSTM layers (Srivastava et al., 2014). We include a dropout layer with p=0.5 before the logistic regression classifier.

C.2 CNN Classifier

We follow the approach of Kim (2014) and train a CNN classifier with kernels of size 3, 4, and 5. We use 100 kernels of each size. We add a dropout layer with p=0.5 between the word embedding layer and the convolutional kernel layers. We include a dropout layer with p=0.5 before the logistic regression classifier.

C.3 REN Classifier

We use the same implementation as Henaff et al. (2017) except that we remove the output module designed to encode questions and instead select the memory cell tied to the entity of interest for every training example. All equations from the input encoder and dynamic memory are identical to those of (Henaff et al., 2017). The input encoder computes a positionally weighted average of all the words in a sentence:

$$s_t = \sum_i f_i \odot e_i \tag{10}$$

where e_i is a word embedding at index i in a sentence, f_i is a positional weight for that index in the sentence, and s_t is a sentence representation. The dynamic memory is updated in the following way:

$$g_j = \sigma(s_t^T h_j + s_t^T w_j) \tag{11}$$

$$\tilde{h}_j = \phi(U h_j + V w_j + W s_t) \tag{12}$$

$$h_j \leftarrow h_j + g_j \odot \tilde{h}_j \tag{13}$$

$$h_j \leftarrow \frac{h_j}{||h_j||} \tag{14}$$

where h_j is the value stored in memory cell j, w_j is a key corresponding to memory cell j, U, V, W are projection matrices, and ϕ is a PReLU nonlinearity. We initialize entity memory keys and entity memory values with the sum of the Glove vectors for all the words in the character name. Entity key values w_j are locked during training. We use dropout with p=0.3 between the encoder and dynamic memory.

C.4 NPN Classifier

We use the same implementation as in Bosselut et al. (2018) with a few modifications to the underlying architecture. First, we use the same encoder

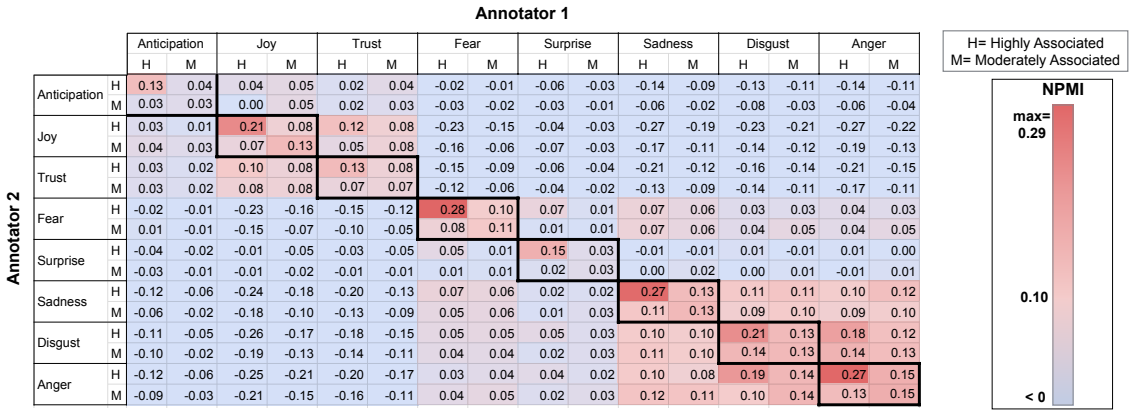

Figure 7: NPMI confusion matrix on emotion categories for all annotator pairs with color scaling for legibility.

as for the REN (Henaff et al., 2017). We define a set of action function embeddings that can be applied to entities to change their state, $\mathbf{A}$. After each sentence, the model selects an action function embedding to use to change the state of the entity memory. Unlike in Bosselut et al. (2018), these action function embeddings are not tied to real actions and are instead treated as latent t The dynamic memory is updated in the following way:

$$g_j = \sigma(s_t^T W_1[h_j, w_j]) \quad (15)$$

$$\alpha_t = softmax(MLP(s_t)) \quad (16)$$

$$a_t = \alpha_t^T \mathbf{A} \quad (17)$$

$$\tilde{h}_j = \phi(W_3 a_t + W_4 s_t) \quad (18)$$

$$h_j \leftarrow (1 - g_j)h_j + g_j \odot \tilde{h}_j \quad (19)$$

$$h_j \leftarrow \frac{h_j}{||h_j||} \quad (20)$$

where h_j is the value stored in memory cell j, w_j is a key corresponding to memory cell j, W_k are projection matrices, ϕ is a PReLU nonlinearity and α_t is a distribution over possible action function embeddings. We initialize entity memory keys and entity memory values with the sum of the Glove vectors for all the words in the character name. Entity key values w_j are locked during training. We use dropout with p=0.5 between the encoder and dynamic memory.

C.5 LSTM Decoder

For the explanation generation task, we train a single-layer LSTM with a learning rate of 0.0003 and gradient clipping when the norm of the gradients exceeds 1. When outputting words, we concatenate the original hidden state from the encoder $\mathbf{h}^e$ to the output of the LSTM decoder before predicting a word. Word embeddings are initialized with pretrained Glove vectors (Pennington et al., 2014). In the generation task, the model must learn to generate individual annotations.

D Experimental Details

D.1 Data used for Annotation Classification Task

We split the development set into two parts, 80% used for training (D_1), 20% used for evaluating hyperparameters (D_2). We train a set of TF-IDF features for each word using all of the explanations from the real training set (D_t) and the portion of the development set used for training (D_1). We train a logistic regression classifier with L2 regularization. When training the classifier on D_1, we only train on examples where the explanation was written by a Turker who selected at least one Plutchik category label that was part of the majority set of Plutchik labels for the sentence the explanation and labels belong to. We prune D_2 and the test set similarly. We use individual annotations rather than majority labels to better learn specific fine-grained mappings.

Emotional Explanation to Plutchik Labels We re-balance the dataset by sampling from the training set evenly among positive examples and negative examples for each category. We use L2 regularization with $\lambda = 0.1$ We include the full positive class distributions for each category in Table 6.

Motivation Explanation to Maslow Labels We use L2 regularization with $\lambda = 0.01$. The dataset is not rebalanced.

Motivation Explanation to Reiss Labels We use L2 regularization with $\lambda = 0.1$. The dataset is not rebalanced.

Maslow		Plutchik		Reiss			
Spiritual growth	6.24	Disgust	5.06	Status	2.53	Romance	2.00
Esteem	8.41	Surprise	11.99	Idealism	0.55	Savings	2.41
Love	11.35	Anger	5.78	Power	1.01	Contact	3.81
Stability	10.46	Trust	9.14	Family	3.57	Health	1.74
Physiological	4.37	Sadness	7.82	Food	2.87	Serenity	0.58
		Anticipation	16.65	Independence	1.29	Curiosity	2.58
		Joy	15.8	Belonging	0.14	Approval	1.88
		Fear	7.15	Competition	2.58	Rest	0.71
				Honor	0.67	Tranquility	2.34
						Order	2.56

Table 6: Class Distribution (percent positive instances) per category.

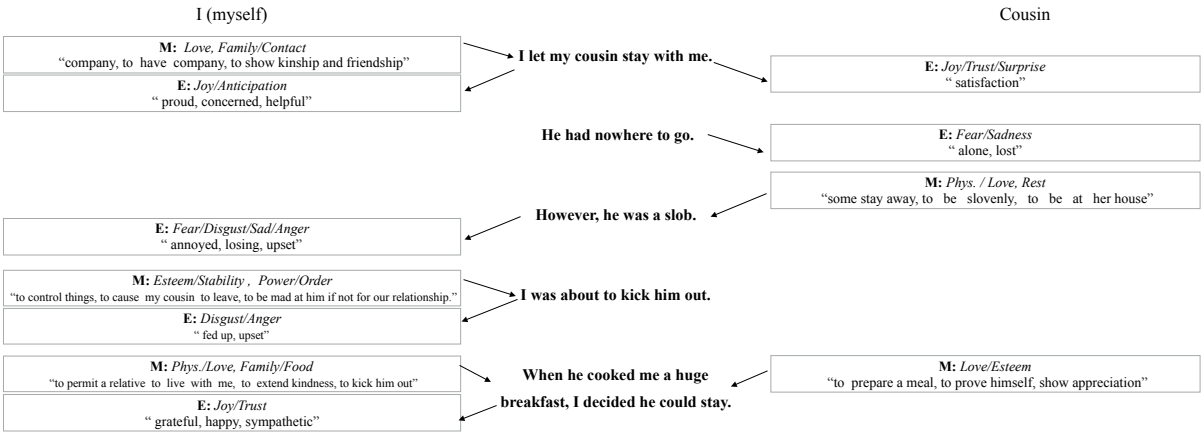

Figure 8: Fully annotated example from the annotation pipeline