A1 Generating machine rationales

Collecting rationales at scale is expensive. Thus, we employ an existing rationalization model (Lei et al., 2016) to generate high-quality rationales automatically for the resource-rich source tasks.

The rationalization model is composed of two modular components: a generator and a classifier. The generator generates rationales from the input, and the classifier maps the generated rationales into the final label. The two components are trained jointly to minimize a loss function that favors short, concise rationales while enforcing that the rationales alone suffice for accurate prediction.

Figure 8 illustrates the model architecture. For the generator, we use a 200 dimensional bi-LSTM to encode the word embedding sequence. Then we apply a linear regressor at each position i to predict the probability p_i that the current word is a rationale. We sample z_i from this probability and pass the sampled rationales $z_i x_i$ to the classifier module. To encourage fast and stable convergence, we use the Gumbel trick (Jang et al., 2016) during sampling.

For the classifier, we employ a CNN-based model (Kim, 2014). Specifically, the classifier first computes 1D convolution over the embeddings of the generated rationales. The filter windows are 3, 5, 7 with 50 feature maps each. It then applies max-over-time pooling to obtain a fixed-length feature vector. Finally, a multilayer perceptron (MLP) is used to predict the label from the feature vector. The MLP consists of one hidden layer (50 units and ReLU activation).

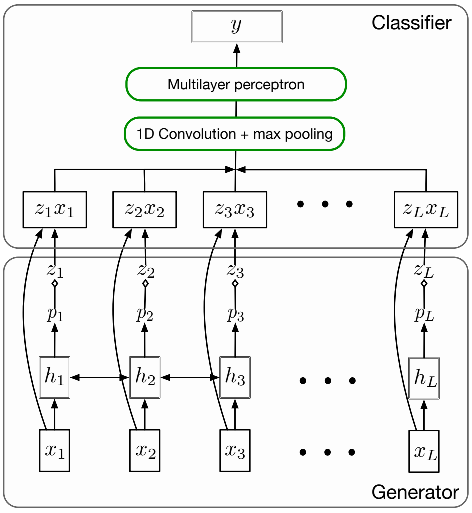
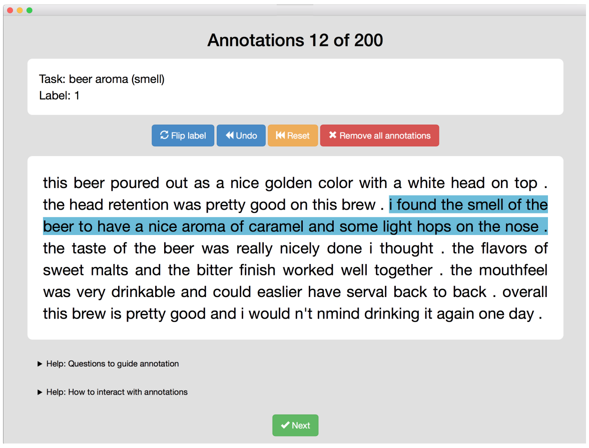


Figure 8: Rationalization model (Lei et al., 2016). Arrow line denotes deterministic computations, while diamond line denotes stochastic sampling.



Annotations 12 of 200

Task: beer aroma (smell)
Label: 1

this beer poured out as a nice golden color with a white head on top . the head retention was pretty good on this brew . i found the smell of the beer to have a nice aroma of caramel and some light hops on the nose . the taste of the beer was really nicely done i thought . the flavors of sweet malts and the bitter finish worked well together . the mouthfeel was very drinkable and could easlier have serval back to back . overall this brew is pretty good and i would n't nmind drinking it again one day .

▸ Help: Questions to guide annotation
▸ Help: How to interact with annotations
✔ Next

Figure 9: Screenshot of the annotation tool

Task	#tokens per review	#rationales per review	#unique tokens	#unique rationales
Beer look	125.4	21.4	35,025	6,460
Beer aroma	126.3	17.6	35,088	5,611
Beer palate	133.5	13.0	36,786	3,885
Hotel location	157.1	13.4	42,717	4,174
Hotel cleanliness	149.4	13.2	42,045	4,758
Hotel service	154.5	17.5	42,989	6,110

Table 7: Statistics of the annotated rationales.

A2 Collecting human rationales

We collected human rationales on six sentiment classification tasks: beer look, beer aroma, beer palate, hotel location, hotel cleanliness and hotel service. For each task, we randomly picked 100 positive examples and 100 negative examples. These 200 labeled examples are given to the annotators (five students) to highlight rationales that are short and coherent, yet sufficient for supporting the label (Lei et al., 2016). The annotators can also flip the original label if it is incorrect. Figure 9 shows our annotation interface, and Table 7 presents the statistics of the collected rationales.

A3 Deriving the oracle attention

Figure 10 illustrates the model architecture that we used to derive the oracle attention. It has the same architecture as the source classifier in our R2A model (Section 3.1). Specifically, we use pre-trained fastText embeddings. The encoder is a bi-LSTM with 200 hidden units. The attention head q is a vector of dimension 50. For the prediction module **pred**, we use a MLP with one hidden

	Beer			Hotel		
	Look	Aroma	Palate	Location	Cleanliness	Service
#training data	32,276	28,984	25,748	14,472	150,098	101,484
#testing data	4,014	4,212	3,804	1,808	12,684	18,762
Testing accuracy	87.17	86.35	82.02	92.20	94.62	95.17

Table 8: Data used to derive the oracle attention. The accuracy is evaluated on the testing data shown in Table 1 and 2.

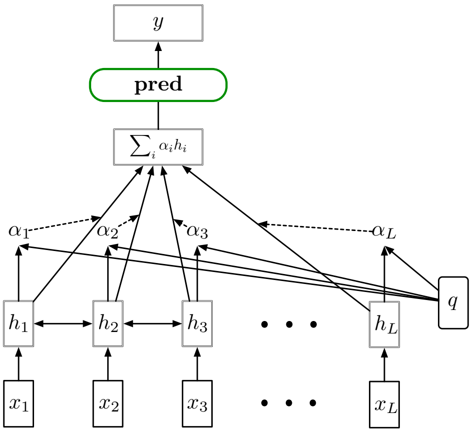


Figure 10: Attention-based classifier

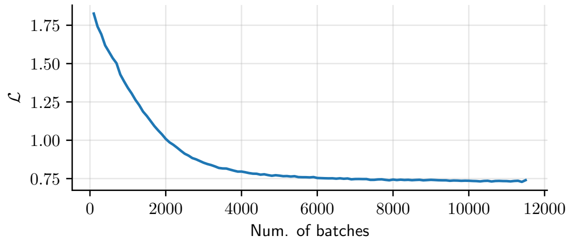


Figure 11: Domain transfer from beer review to hotel review. Overall loss $\mathcal{L}$ of R2A on the source development set during training.

layer (50 units and ReLU activation). For each task, we train this attention-based classifier on a large amount of annotations. Table 8 presents the statistics of the training data and the testing performance of the classifier.

A4 Estimating the Wasserstein distance

We train a critic network to estimate the Wasserstein distance between the distribution of the representation from the source domain and the one from the target domain:

$$\mathcal{L}_{wd} = \sup_{\|f\|_L \le K} \mathbb{E}_{h^{\mathbf{inv}} \sim \mathbb{P}_\mathcal{S}} \left[f([h_1^{\mathbf{inv}}; h_L^{\mathbf{inv}}]) \right] - \mathbb{E}_{h^{\mathbf{inv}} \sim \mathbb{P}_\mathcal{T}} \left[f([h_1^{\mathbf{inv}}; h_L^{\mathbf{inv}}]) \right].$$

The critic network is parametrized as a MLP with one hidden layer (100 units and ReLU activation). Following Gulrajani et al. (2017), we set the weight of the gradient penalty to 10 and optimize the critic network for 5 iterations during each batch. Figure 11 plots the loss on the development set versus the number of training batches. We see that our R2A model converges smoothly during training.