Interpretable Neural Predictions with Differentiable Binary Variables

Jasmijn Bastings
ILLC
University of Amsterdam
bastings@uva.nl

Wilker Aziz
ILLC
University of Amsterdam
w.aziz@uva.nl

Ivan Titov
ILLC, University of Amsterdam
ILCC, University of Edinburgh
ititov@inf.ed.ac.uk

Abstract

The success of neural networks comes hand in hand with a desire for more interpretability. We focus on text classifiers and make them more interpretable by having them provide a justification—a *rationale*—for their predictions. We approach this problem by jointly training two neural network models: a latent model that selects a rationale (i.e. a short and informative part of the input text), and a classifier that learns from the words in the rationale alone. Previous work proposed to assign binary latent masks to input positions and to promote short selections via sparsity-inducing penalties such as L_0 regularisation. We propose a latent model that mixes discrete and continuous behaviour allowing at the same time for binary selections and gradient-based training without REINFORCE. In our formulation, we can tractably compute the expected value of penalties such as L_0, which allows us to directly optimise the model towards a pre-specified text selection rate. We show that our approach is competitive with previous work on rationale extraction, and explore further uses in attention mechanisms.

1 Introduction

Neural networks are bringing incredible performance gains on text classification tasks (Howard and Ruder, 2018; Peters et al., 2018; Devlin et al., 2019). However, this power comes hand in hand with a desire for more interpretability, even though its definition may differ (Lipton, 2016). While it is useful to obtain high classification accuracy, with more data available than ever before it also becomes increasingly important to *justify* predictions. Imagine having to classify a large collection of documents, while verifying that the classifications make sense. It would be extremely time-consuming to read each document to evaluate the results. Moreover, if we do not know why a prediction was made, we do not know if we can trust it.

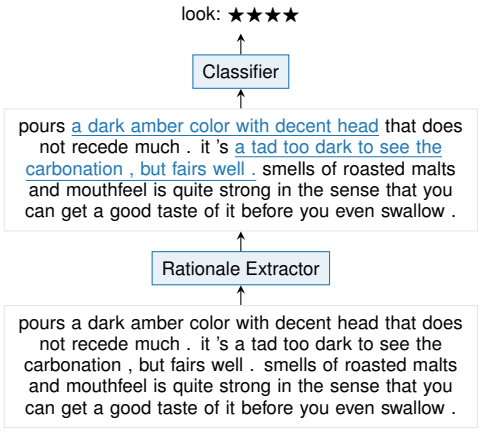


Figure 1: Rationale extraction for a beer review.

What if the model could provide us the most important parts of the document, as a justification for its prediction? That is exactly the focus of this paper. We use a setting that was pioneered by Lei et al. (2016). A *rationale* is defined to be a *short* yet *sufficient* part of the input text; short so that it makes clear what is most important, and sufficient so that a correct prediction can be made from the rationale alone. One neural network learns to extract the rationale, while another neural network, with separate parameters, learns to make a prediction from just the rationale. Lei et al. model this by assigning a binary Bernoulli variable to each input word. The rationale then consists of all the words for which a 1 was sampled. Because gradients do not flow through discrete samples, the rationale extractor is optimized using REINFORCE (Williams, 1992). An L_0 regularizer is used to make sure the rationale is short.

We propose an alternative to purely discrete selectors for which gradient estimation is possible without REINFORCE, instead relying on a repa-

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 2963–2977
Florence, Italy, July 28 - August 2, 2019.

rameterization of a random variable that exhibits both continuous and discrete behavior (Louizos et al., 2017). To promote compact rationales, we employ a relaxed form of L_0 regularization (Louizos et al., 2017), penalizing the objective as a function of the expected proportion of selected text. We also propose the use of Lagrangian relaxation to target a specific rate of selected input text.

Our contributions are summarized as follows:[1]

1. we present a differentiable approach to extractive rationales (§2) including an objective that allows for specifying how much text is to be extracted (§4);

2. we introduce HardKuma (§3), which gives support to binary outcomes and allows for reparameterized gradient estimates;

3. we empirically show that our approach is competitive with previous work and that HardKuma has further applications, e.g. in attention mechanisms. (§6).

2 Latent Rationale

We are interested in making NN-based text classifiers interpretable by (i) uncovering which parts of the input text contribute features for classification, and (ii) basing decisions on only a fraction of the input text (a *rationale*). Lei et al. (2016) approached (i) by inducing binary latent selectors that control which input positions are available to an NN encoder that learns features for classification/regression, and (ii) by regularising their architectures using sparsity-inducing penalties on latent assignments. In this section we put their approach under a probabilistic light, and this will then more naturally lead to our proposed method.

In text classification, an input x is mapped to a distribution over target labels:

$$Y|x \sim \text{Cat}(f(x;\theta)) \,, \tag{1}$$

where we have a neural network architecture $f(\cdot;\theta)$ parameterize the model—θ collectively denotes the parameters of the NN layers in f. That is, an NN maps from data space (e.g. sentences, short paragraphs, or premise-hypothesis pairs) to the categorical parameter space (i.e. a vector of class probabilities). For the sake of concreteness, consider the input a sequence $x = \langle x_1, \ldots, x_n \rangle$. A target y is typically a categorical outcome, such as a sentiment class or an entailment decision, but with an appropriate choice of likelihood it could also be a numerical score (continuous or integer).

Lei et al. (2016) augment this model with a collection of latent variables which we denote by $z = \langle z_1, \ldots, z_n \rangle$. These variables are responsible for regulating which portions of the input x contribute with predictors (i.e. features) to the classifier. The model formulation changes as follows:

$$\begin{aligned} Z_i|x &\sim \text{Bern}(g_i(x;\phi)) \\ Y|x,z &\sim \text{Cat}(f(x \odot z;\theta)) \end{aligned} \tag{2}$$

where an NN $g(\cdot;\phi)$ predicts a sequence of n Bernoulli parameters—one per latent variable—and the classifier is modified such that z_i indicates whether or not x_i is available for encoding. We can think of the sequence z as a binary gating mechanism used to select a rationale, which with some abuse of notation we denote by $x \odot z$. Figure 1 illustrates the approach.

Parameter estimation for this model can be done by maximizing a lower bound $\mathcal{E}(\phi,\theta)$ on the log-likelihood of the data derived by application of Jensen's inequality:[2]

$$\begin{aligned} &\log P(y|x) = \log \mathbb{E}_{P(z|x,\phi)}\left[P(y|x,z,\theta)\right] \\ &\overset{\text{JI}}{\geq} \mathbb{E}_{P(z|x,\phi)}\left[\log P(y|x,z,\theta)\right] = \mathcal{E}(\phi,\theta) \,. \end{aligned} \tag{3}$$

These latent rationales approach the first objective, namely, uncovering which parts of the input text contribute towards a decision. However note that an NN controls the Bernoulli parameters, thus nothing prevents this NN from selecting the whole of the input, thus defaulting to a standard text classifier. To promote compact rationales, Lei et al. (2016) impose sparsity-inducing penalties on latent selectors. They penalise for the total number of selected words, L_0 in (4), as well as, for the total number of transitions, *fused lasso* in (4), and approach the following optimization problem

$$\min_{\phi,\theta} -\mathcal{E}(\phi,\theta) + \lambda_0 \underbrace{\sum_{i=1}^{n} z_i}_{L_0(z)} + \lambda_1 \underbrace{\sum_{i=1}^{n-1} |z_i - z_{i+1}|}_{\text{fused lasso}} \tag{4}$$

via gradient-based optimisation, where λ_0 and λ_1 are fixed hyperparameters. The objective is however intractable to compute, the lowerbound, in

[1] Code available at https://github.com/bastings/interpretable_predictions.

[2] This can be seen as variational inference (Jordan et al., 1999) where we perform approximate inference using a data-dependent prior $P(z|x,\phi)$.

particular, requires marginalization of $O(2^n)$ binary sequences. For that reason, Lei et al. sample latent assignments and work with gradient estimates using REINFORCE (Williams, 1992).

The key ingredients are, therefore, binary latent variables and sparsity-inducing regularization, and therefore the solution is marked by non-differentiability. We propose to replace Bernoulli variables by rectified continuous random variables (Socci et al., 1998), for they exhibit both discrete and continuous behaviour. Moreover, they are amenable to reparameterization in terms of a fixed random source (Kingma and Welling, 2014), in which case gradient estimation is possible without REINFORCE. Following Louizos et al. (2017), we exploit one such distribution to relax L_0 regularization and thus promote compact rationales with a differentiable objective. In section 3, we introduce this distribution and present its properties. In section 4, we employ a Lagrangian relaxation to automatically target a pre-specified selection rate. And finally, in section 5 we present an example for sentiment classification.

3 Hard Kumaraswamy Distribution

Key to our model is a novel distribution that exhibits both continuous and discrete behaviour, in this section we introduce it. With non-negligible probability, samples from this distribution evaluate to exactly 0 or exactly 1. In a nutshell: i) we start from a distribution over the open interval $(0, 1)$ (see dashed curve in Figure 2); ii) we then *stretch* its support from $l < 0$ to $r > 1$ in order to include $\{0\}$ and $\{1\}$ (see solid curve in Figure 2); finally, iii) we collapse the probability mass over the interval $(l, 0]$ to $\{0\}$, and similarly, the probability mass over the interval $[1, r)$ to $\{1\}$ (shaded areas in Figure 2). This *stretch-and-rectify* technique was proposed by Louizos et al. (2017), who rectified samples from the BinaryConcrete (or GumbelSoftmax) distribution (Maddison et al., 2017; Jang et al., 2017). We adapted their technique to the Kumaraswamy distribution motivated by its close resemblance to a Beta distribution, for which we have stronger intuitions (for example, its two shape parameters transit rather naturally from unimodal to bimodal configurations of the distribution). In the following, we introduce this new distribution formally.[3]

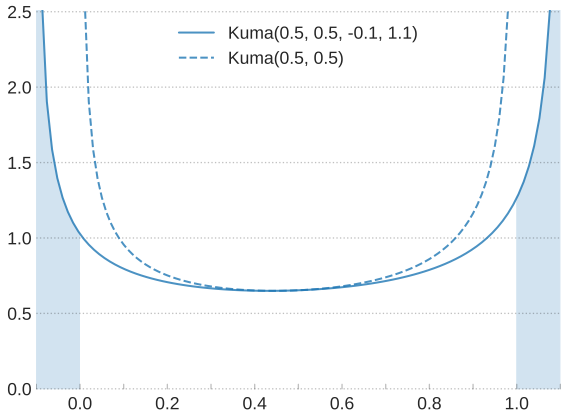


Figure 2: The HardKuma distribution: we start from a $\mathrm{Kuma}(0.5, 0.5)$, and stretch its support to the interval $(-0.1, 1.1)$, finally we collapse all mass before 0 to $\{0\}$ and all mass after 1 to $\{1\}$.

3.1 Kumaraswamy distribution

The Kumaraswamy distribution (Kumaraswamy, 1980) is a two-parameters distribution over the **open** interval $(0, 1)$, we denote a Kumaraswamy-distributed variable by $K \sim \mathrm{Kuma}(a, b)$, where $a \in \mathbb{R}_{>0}$ and $b \in \mathbb{R}_{>0}$ control the distribution's shape. The dashed curve in Figure 2 illustrates the density of $\mathrm{Kuma}(0.5, 0.5)$. For more details including its pdf and cdf, consult Appendix A.

The Kumaraswamy is a close relative of the Beta distribution, though not itself an exponential family, with a simple cdf whose inverse

$$F_K^{-1}(u; a, b) = \left(1 - (1 - u)^{1/b}\right)^{1/a} , \quad (5)$$

for $u \in [0, 1]$, can be used to obtain samples

$$F_K^{-1}(U; \alpha, \beta) \sim \mathrm{Kuma}(\alpha, \beta) \quad (6)$$

by transformation of a uniform random source $U \sim \mathcal{U}(0, 1)$. We can use this fact to reparameterize expectations (Nalisnick and Smyth, 2016).

3.2 Rectified Kumaraswamy

We *stretch* the support of the Kumaraswamy distribution to include 0 and 1. The resulting variable $T \sim \mathrm{Kuma}(a, b, l, r)$ takes on values in the open interval (l, r) where $l < 0$ and $r > 1$, with cdf

$$F_T(t; a, b, l, r) = F_K((t - l)/(r - l); a, b) . \quad (7)$$

We now define a rectified random variable, denoted by $H \sim \mathrm{HardKuma}(a, b, l, r)$, by passing

[3] We use uppercase letters for random variables (e.g. K, T, and H) and lowercase for assignments (e.g. k, t, h). For a random variable K, $f_K(k; \alpha)$ is the probability density function (pdf), conditioned on parameters α, and $F_K(k; \alpha)$ is the cumulative distribution function (cdf).

a sample $T \sim \text{Kuma}(a, b, l, r)$ through a hard-sigmoid, i.e. $h = \min(1, \max(0, t))$. The resulting variable is defined over the **closed** interval $[0, 1]$. Note that while there is 0 probability of sampling $t = 0$, sampling $h = 0$ corresponds to sampling any $t \in (l, 0]$, a set whose mass under $\text{Kuma}(t|a, b, l, r)$ is available in closed form:

$$\mathbb{P}(H = 0) = F_K\left(\tfrac{-l}{r-l}; a, b\right) . \tag{8}$$

That is because all negative values of t are deterministically mapped to zero. Similarly, samples $t \in [1, r)$ are all deterministically mapped to $h = 1$, whose total mass amounts to

$$\mathbb{P}(H = 1) = 1 - F_K\left(\tfrac{1-l}{r-l}; a, b\right) . \tag{9}$$

See Figure 2 for an illustration, and Appendix A for the complete derivations.

3.3 Reparameterization and gradients

Because this rectified variable is built upon a Kumaraswamy, it admits a reparameterisation in terms of a uniform variable $U \sim \mathcal{U}(0, 1)$. We need to first sample a uniform variable in the open interval $(0, 1)$ and transform the result to a Kumaraswamy variable via the inverse cdf (10a), then shift and scale the result to cover the stretched support (10b), and finally, apply the rectifier in order to get a sample in the closed interval $[0, 1]$ (10c).

$$k = F_K^{-1}(u; a, b) \tag{10a}$$
$$t = l + (r - l)k \tag{10b}$$
$$h = \min(1, \max(0, t)) , \tag{10c}$$

We denote this $h = s(u; a, b, l, r)$ for short. Note that this transformation has two discontinuity points, namely, $t = 0$ and $t = 1$. Though recall, the probability of sampling t exactly 0 or exactly 1 is zero, which essentially means stochasticity circumvents points of non-differentiability of the rectifier (see Appendix A.3).

4 Controlled Sparsity

Following Louizos et al. (2017), we relax non-differentiable penalties by computing them on expectation under our latent model $p(z|x, \phi)$. In addition, we propose the use of Lagrangian relaxation to target specific values for the penalties. Thanks to the tractable Kumaraswamy cdf, the expected value of $L_0(z)$ is known in closed form

$$\begin{aligned}\mathbb{E}_{p(z|x)}\left[L_0(z)\right] &\overset{\text{ind}}{=} \sum_{i=1}^{n} \mathbb{E}_{p(z_i|x)}\left[\mathbb{I}[z_i \neq 0]\right] \\ &= \sum_{i=1}^{n} 1 - \mathbb{P}(Z_i = 0) ,\end{aligned} \tag{11}$$

where $\mathbb{P}(Z_i = 0) = F_K\left(\tfrac{-l}{r-l}; a_i, b_i\right)$. This quantity is a tractable and differentiable function of the parameters ϕ of the latent model. We can also compute a relaxation of fused lasso by computing the expected number of zero-to-nonzero and nonzero-to-zero changes:

$$\begin{aligned}&\mathbb{E}_{p(z|x)}\left[\sum_{i=1}^{n-1} \mathbb{I}[z_i = 0, z_{i+1} \neq 0]\right] \\ &+ \mathbb{E}_{p(z|x)}\left[\sum_{i=1}^{n-1} \mathbb{I}[z_i \neq 0, z_{i+1} = 0]\right] \\ &\overset{\text{ind}}{=} \sum_{i=1}^{n-1} \mathbb{P}(Z_i = 0)(1 - \mathbb{P}(Z_{i+1} = 0)) \\ &+ (1 - \mathbb{P}(Z_i = 0))\mathbb{P}(Z_{i+1} = 0) .\end{aligned} \tag{12}$$

In both cases, we make the assumption that latent variables are independent given x, in Appendix B.1.2 we discuss how to estimate the regularizers for a model $p(z_i|x, z_{<i})$ that conditions on the prefix $z_{<i}$ of sampled HardKuma assignments.

We can use regularizers to promote sparsity, but just how much text will our final model select? Ideally, we would target specific values r and solve a constrained optimization problem. In practice, constrained optimisation is very challenging, thus we employ Lagrangian relaxation instead:

$$\max_{\lambda \in \mathbb{R}} \min_{\phi, \theta} -\mathcal{E}(\phi, \theta) + \lambda^\top (R(\phi) - r) \tag{13}$$

where $R(\phi)$ is a vector of regularisers, e.g. expected L_0 and expected fused lasso, and λ is a vector of Lagrangian multipliers λ. Note how this differs from the treatment of Lei et al. (2016) shown in (4) where regularizers are computed for assignments, rather than on expectation, and where λ_0, λ_1 are fixed hyperparameters.

5 Sentiment Classification

As a concrete example, consider the case of sentiment classification where x is a sentence and y is a

5-way sentiment class (from very negative to very positive). The model consists of

$$\begin{aligned} Z_i &\sim \text{HardKuma}(a_i, b_i, l, r) \\ Y|x, z &\sim \text{Cat}(f(x \odot z; \theta)) \end{aligned} \tag{14}$$

where the shape parameters $a, b = g(x; \phi)$, i.e. two sequences of n strictly positive scalars, are predicted by a NN, and the support boundaries (l, r) are fixed hyperparameters.

We first specify an architecture that parameterizes latent selectors and then use a reparameterized sample to restrict which parts of the input contribute encodings for classification:[4]

$$\begin{aligned} \mathbf{e}_i &= \text{emb}(x_i) & a_i &= f_a(\mathbf{h}_i; \phi_a) \\ \mathbf{h}_1^n &= \text{birnn}(\mathbf{e}_1^n; \phi_r) & b_i &= f_b(\mathbf{h}_i; \phi_b) \\ u_i &\sim \mathcal{U}(0, 1) & z_i &= s(u_i; a_i, b_i, l, r) \end{aligned}$$

where $\text{emb}(\cdot)$ is an embedding layer, $\text{birnn}(\cdot; \phi_r)$ is a bidirectional encoder, $f_a(\cdot; \phi_a)$ and $f_b(\cdot; \phi_b)$ are feed-forward transformations with softplus outputs, and $s(\cdot)$ turns the uniform sample u_i into the latent selector z_i (see §3). We then use the sampled z to modulate inputs to the classifier:

$$\begin{aligned} \mathbf{e}_i &= \text{emb}(x_i) \\ \mathbf{h}_i^{(\text{fwd})} &= \text{rnn}(\mathbf{h}_{i-1}^{(\text{fwd})}, z_i\, \mathbf{e}_i; \theta_{\text{fwd}}) \\ \mathbf{h}_i^{(\text{bwd})} &= \text{rnn}(\mathbf{h}_{i+1}^{(\text{bwd})}, z_i\, \mathbf{e}_i; \theta_{\text{bwd}}) \\ \mathbf{o} &= f_o(\mathbf{h}_n^{(\text{fwd})}, \mathbf{h}_1^{(\text{bwd})}; \theta_o) \end{aligned}$$

where $\text{rnn}(\cdot; \theta_{\text{fwd}})$ and $\text{rnn}(\cdot; \theta_{\text{bwd}})$ are recurrent cells such as LSTMs (Hochreiter and Schmidhuber, 1997) that process the sequence in different directions, and $f_o(\cdot; \theta_o)$ is a feed-forward transformation with softmax output. Note how z_i modulates features $\mathbf{e}_i$ of the input x_i that are available to the recurrent composition function.

We then obtain gradient estimates of $\mathcal{E}(\phi, \theta)$ via Monte Carlo (MC) sampling from

$$\mathcal{E}(\phi, \theta) = \mathbb{E}_{\mathcal{U}(0,I)} \left[\log P(y|x, s_\phi(u, x), \theta)\right] \tag{15}$$

where $z = s_\phi(u, x)$ is a shorthand for element-wise application of the transformation from uniform samples to HardKuma samples. This reparameterisation is the key to gradient estimation through stochastic computation graphs (Kingma and Welling, 2014; Rezende et al., 2014).

[4] We describe architectures using blocks denoted by layer(inputs; subset of parameters), boldface letters for vectors, and the shorthand $\mathbf{v}_1^n$ for a sequence $\langle \mathbf{v}_1, \ldots, \mathbf{v}_n \rangle$.

SVM (Lei et al., 2016)	0.0154
BiLSTM (Lei et al., 2016)	0.0094
BiRCNN (Lei et al., 2016)	0.0087
BiLSTM *(ours)*	0.0089
BiRCNN *(ours)*	0.0088

Table 1: MSE on the BeerAdvocate test set.

Deterministic predictions. At test time we make predictions based on what is the most likely assignment for each z_i. We $\arg\max$ across configurations of the distribution, namely, $z_i = 0$, $z_i = 1$, or $0 < z_i < 1$. When the continuous interval is more likely, we take the expected value of the underlying Kumaraswamy variable.

6 Experiments

We perform experiments on multi-aspect sentiment analysis to compare with previous work, as well as experiments on sentiment classification and natural language inference. All models were implemented in PyTorch, and Appendix B provides implementation details.

Goal. When rationalizing predictions, our goal is to perform as well as systems using the full input text, while using only a *subset* of the input text, leaving unnecessary words out for interpretability.

6.1 Multi-aspect Sentiment Analysis

In our first experiment we compare directly with previous work on rationalizing predictions (Lei et al., 2016). We replicate their setting.

Data. A pre-processed subset of the BeerAdvocate[5] data set is used (McAuley et al., 2012). It consists of 220,000 *beer reviews*, where multiple aspects (e.g. look, smell, palate) are rated. As shown in Figure 1, a review typically consists of multiple sentences, and contains a 0-5 star rating (e.g. 3.5 stars) for each aspect. Lei et al. mapped the ratings to scalars in $[0, 1]$.

Model. We use the models described in §5 with two small modifications: 1) since this is a regression task, we use a sigmoid activation in the output layer of the classifier rather than a softmax,[6] and

[5] https://www.beeradvocate.com/

[6] From a likelihood learning point of view, we would have assumed a Logit-Normal likelihood, however, to stay closer to Lei et al. (2016), we employ mean squared error.

Method	Look		Smell		Palate	
	% Precision	% Selected	% Precision	% Selected	% Precision	% Selected
Attention (Lei et al.)	80.6	13	88.4	7	65.3	7
Bernoulli (Lei et al.)	96.3	14	95.1	7	80.2	7
Bernoulli *(reimpl.)*	94.8	13	95.1	7	80.5	7
HardKuma	98.1	13	96.8	7	89.8	7

Table 2: Precision (% of selected words that was also annotated as the gold rationale) and selected (% of words not zeroed out) per aspect. In the attention baseline, the top 13% (7%) of words with highest attention weights are used for classification. Models were selected based on validation loss.

2) we use an extra RNN to condition z_i on $z_{<i}$:

$$a_i = f_a(\mathbf{h}_i, \mathbf{s}_{i-1}; \phi_a) \quad (16a)$$
$$b_i = f_b(\mathbf{h}_i, \mathbf{s}_{i-1}; \phi_b) \quad (16b)$$
$$\mathbf{s}_i = \text{rnn}(\mathbf{h}_i, z_i, \mathbf{s}_{i-1}; \phi_s) \quad (16c)$$

For a fair comparison we follow Lei et al. by using RCNN[7] cells rather than LSTM cells for encoding sentences on this task. Since this cell is not widely used, we verified its performance in Table 1. We observe that the BiRCNN performs on par with the BiLSTM (while using 50% fewer parameters), and similarly to previous results.

Evaluation. A test set with sentence-level rationale annotations is available. The *precision* of a rationale is defined as the percentage of words with $z \neq 0$ that is part of the annotation. We also evaluate the predictions made from the rationale using mean squared error (MSE).

Baselines. For our baseline we reimplemented the approach of Lei et al. (2016) which we call *Bernoulli* after the distribution they use to sample z from. We also report their attention baseline, in which an attention score is computed for each word, after which it is simply thresholded to select the top-k percent as the rationale.

Results. Table 2 shows the precision and the percentage of selected words for the first three aspects. The models here have been selected based on validation MSE and were tuned to select a similar percentage of words ('selected'). We observe that our Bernoulli reimplementation reaches the precision similar to previous work, doing a little bit worse for the 'look' aspect. Our HardKuma managed to get even higher precision, and it extracted exactly the percentage of text that we specified (see §4).[8] Figure 3 shows the MSE for all aspects for various percentages of extracted text. We observe that HardKuma does better with a smaller percentage of text selected. The performance becomes more similar as more text is selected.

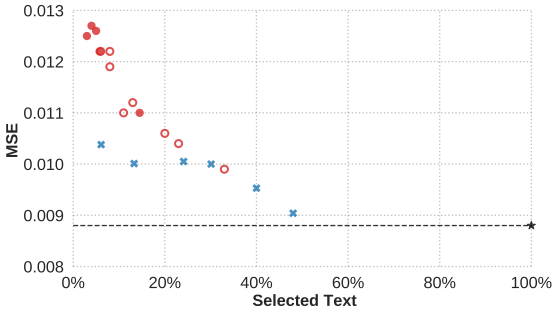


Figure 3: MSE of all aspects for various percentages of extracted text. HardKuma (blue crosses) has lower error than Bernoulli (red circles; open circles taken from Lei et al. (2016)) for similar amount of extracted text. The full-text baseline (black star) gets the best MSE.

6.2 Sentiment Classification

We also experiment on the Stanford Sentiment Treebank (SST) (Socher et al., 2013). There are 5 sentiment classes: very negative, negative, neutral, positive, and very positive. Here we use the HardKuma model described in §5, a Bernoulli model trained with REINFORCE, as well as a BiLSTM.

Results. Figure 4 shows the classification accuracy for various percentages of selected text. We observe that HardKuma outperforms the Bernoulli model at each percentage of selected text. HardKuma reaches full-text baseline performance already around 40% extracted text. At that point, it obtains a *test* score of 45.84, versus 42.22 for Bernoulli and 47.4±0.8 for the full-text baseline.

[7] An RCNN cell can replace any LSTM cell and works well on text classification problems. See appendix B.

[8] We tried to use Lagrangian relaxation for the Bernoulli model, but this led to instabilities (e.g. all words selected).

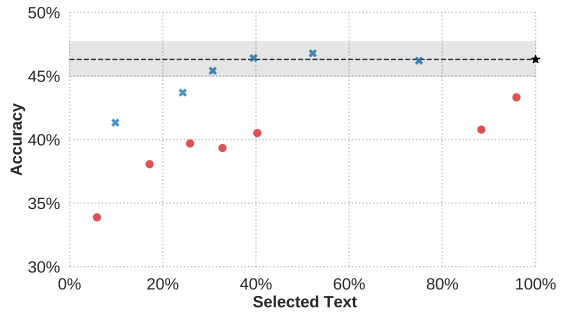


Figure 4: SST validation accuracy for various percentages of extracted text. HardKuma (blue crosses) has higher accuracy than Bernoulli (red circles) for similar amount of text, and reaches the full-text baseline (black star, $46.3 \pm 2\sigma$ with $\sigma = 0.7$) around 40% text.

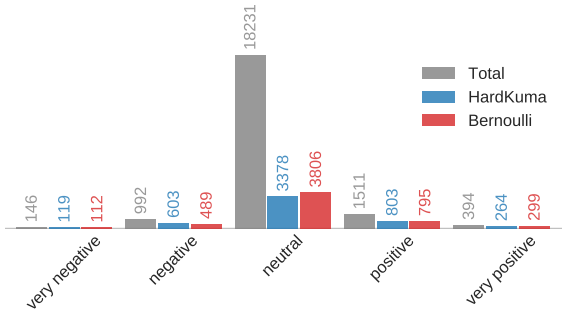


Figure 5: The number of words in each sentiment class for the full validation set, the HardKuma (24% selected text) and Bernoulli (25% text).

Analysis. We wonder what kind of words are dropped when we select smaller amounts of text. For this analysis we exploit the word-level sentiment annotations in SST, which allows us to track the sentiment of words in the rationale. Figure 5 shows that a large portion of dropped words have neutral sentiment, and it seems plausible that exactly those words are not important features for classification. We also see that HardKuma drops (relatively) more neutral words than Bernoulli.

6.3 Natural Language Inference

In Natural language inference (NLI), given a premise sentence $x^{(p)}$ and a hypothesis sentence $x^{(h)}$, the goal is to predict their relation y which can be contradiction, entailment, or neutral. As our dataset we use the Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015).

Baseline. We use the Decomposable Attention model (DA) of Parikh et al. (2016).[9] DA does not make use of LSTMs, but rather uses attention to find connections between the premise and the hypothesis that are predictive of the relation. Each word in the premise attends to each word in the hypothesis, and vice versa, resulting in a set of comparison vectors which are then aggregated for a final prediction. If there is no link between a word pair, it is not considered for prediction.

Model. Because the premise and hypothesis interact, it does not make sense to extract a rationale for the premise and hypothesis independently. Instead, we replace the attention between premise and hypothesis with HardKuma attention. Whereas in the baseline a similarity matrix is $\mathrm{softmax}$-normalized across rows (premise to hypothesis) and columns (hypothesis to premise) to produce attention matrices, in our model each cell in the attention matrix is sampled from a HardKuma parameterized by (a, b). To promote sparsity, we use the relaxed L_0 to specify the desired percentage of non-zero attention cells. The resulting matrix does not need further normalization.

Results. With a target rate of 10%, the HardKuma model achieved 8.5% non-zero attention. Table 3 shows that, even with so many zeros in the attention matrices, it only does about 1% worse compared to the DA baseline. Figure 6 shows an example of HardKuma attention, with additional examples in Appendix B. We leave further explorations with HardKuma attention for future work.

Model	Dev	Test
LSTM (Bowman et al., 2016)	–	80.6
DA (Parikh et al., 2016)	–	86.3
DA *(reimplementation)*	86.9	86.5
DA with HardKuma attention	86.0	85.5

Table 3: SNLI results (accuracy).

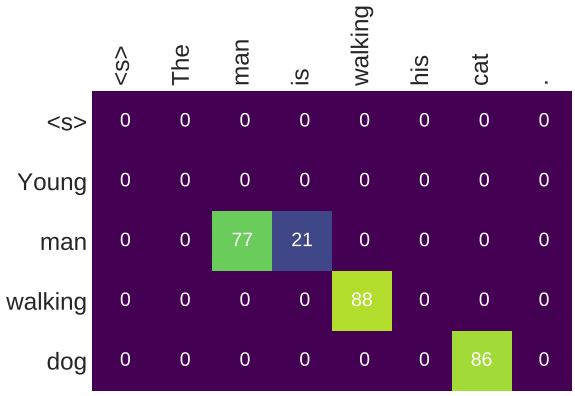


Figure 6: Example of HardKuma attention between a premise (rows) and hypothesis (columns) in SNLI (cell values shown in multiples of 10^{-2}).

[9] Better results e.g. Chen et al. (2017) and data sets for NLI exist, but are not the focus of this paper.

7 Related Work

This work has connections with work on interpretability, learning from rationales, sparse structures, and rectified distributions. We discuss each of those areas.

Interpretability. Machine learning research has been focusing more and more on interpretability (Gilpin et al., 2018). However, there are many nuances to *interpretability* (Lipton, 2016), and amongst them we focus on model transparency.

One strategy is to extract a simpler, interpretable model from a neural network, though this comes at the cost of performance. For example, Thrun (1995) extract if-then rules, while Craven and Shavlik (1996) extract decision trees.

There is also work on making word vectors more interpretable. Faruqui et al. (2015) make word vectors more sparse, and Herbelot and Vecchi (2015) learn to map distributional word vectors to model-theoretic semantic vectors.

Similarly to Lei et al. (2016), Titov and McDonald (2008) extract informative fragments of text by jointly training a classifier and a model predicting a stochastic mask, while relying on Gibbs sampling to do so. Their focus is on using the sentiment labels as a weak supervision signal for opinion summarization rather than on rationalizing classifier predictions.

There are also related approaches that aim to interpret an already-trained model, in contrast to Lei et al. (2016) and our approach where the rationale is jointly modeled. Ribeiro et al. (2016) make any classifier interpretable by approximating it locally with a linear proxy model in an approach called LIME, and Alvarez-Melis and Jaakkola (2017) propose a framework that returns input-output pairs that are causally related.

Learning from rationales. Our work is different from approaches that aim to improve classification using rationales as an additional input (Zaidan et al., 2007; Zaidan and Eisner, 2008; Zhang et al., 2016). Instead, our rationales are latent and we are interested in uncovering them. We only use annotated rationales for evaluation.

Sparse layers. Also arguing for enhanced interpretability, Niculae and Blondel (2017) propose a framework for learning sparsely activated attention layers based on smoothing the $\max$ operator. They derive a number of relaxations to $\max$, including $\mathrm{softmax}$ itself, but in particular, they target relaxations such as $\mathrm{sparsemax}$ (Martins and Astudillo, 2016) which, unlike $\mathrm{softmax}$, are sparse (i.e. produce vectors of probability values with components that evaluate to exactly 0). Their activation functions are themselves solutions to convex optimization problems, to which they provide efficient forward and backward passes. The technique can be seen as a deterministic sparsely activated layer which they use as a drop-in replacement to standard attention mechanisms. In contrast, in this paper we focus on binary outcomes rather than K-valued ones. Niculae et al. (2018) extend the framework to structured discrete spaces where they learn sparse parameterizations of discrete latent models. In this context, parameter estimation requires exact marginalization of discrete variables or gradient estimation via REINFORCE. They show that oftentimes distributions are sparse enough to enable exact marginal inference.

Peng et al. (2018) propose SPIGOT, a proxy gradient to the non-differentiable $\arg\max$ operator. This proxy requires an $\arg\max$ solver (e.g. Viterbi for structured prediction) and, like the straight-through estimator (Bengio et al., 2013), is a biased estimator. Though, unlike ST it is efficient for structured variables. In contrast, in this work we chose to focus on unbiased estimators.

Rectified Distributions. The idea of rectified distributions has been around for some time. The rectified Gaussian distribution (Socci et al., 1998), in particular, has found applications to factor analysis (Harva and Kaban, 2005) and approximate inference in graphical models (Winn and Bishop, 2005). Louizos et al. (2017) propose to stretch and rectify samples from the BinaryConcrete (or GumbelSoftmax) distribution (Maddison et al., 2017; Jang et al., 2017). They use rectified variables to induce sparsity in parameter space via a relaxation to L_0. We adapt their technique to promote sparse *activations* instead. Rolfe (2017) learns a relaxation of a discrete random variable based on a tractable mixture of a point mass at zero and a continuous reparameterizable density, thus enabling reparameterized sampling from the half-closed interval $[0, \infty)$. In contrast, with HardKuma we focused on giving support to both 0s and 1s.

8 Conclusions

We presented a differentiable approach to extractive rationales, including an objective that allows

for specifying how much text is to be extracted. To allow for reparameterized gradient estimates and support for binary outcomes we introduced the HardKuma distribution. Apart from extracting rationales, we showed that HardKuma has further potential uses, which we demonstrated on premise-hypothesis attention in SNLI. We leave further explorations for future work.

Acknowledgments

We thank Luca Falorsi for pointing us to Louizos et al. (2017), which inspired the HardKumaraswamy distribution. This work has received funding from the European Research Council (ERC StG BroadSem 678254), the European Union's Horizon 2020 research and innovation programme (grant agreement No 825299, GoURMET), and the Dutch National Science Foundation (NWO VIDI 639.022.518, NWO VICI 277-89-002).

References

David Alvarez-Melis and Tommi Jaakkola. 2017. A causal framework for explaining the predictions of black-box sequence-to-sequence models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 412–421. Association for Computational Linguistics.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642. Association for Computational Linguistics.

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1466–1477. Association for Computational Linguistics.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668. Association for Computational Linguistics.

Mark Craven and Jude W Shavlik. 1996. Extracting tree-structured representations of trained networks. In *Advances in neural information processing systems*, pages 24–30.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A. Smith. 2015. Sparse overcomplete word vector representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1491–1500. Association for Computational Linguistics.

Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. 2018. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 80–89. IEEE.

Markus Harva and Ata Kaban. 2005. A variational bayesian method for rectified factor analysis. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 1, pages 185–190. IEEE.

Aurélie Herbelot and Eva Maria Vecchi. 2015. Building a shared world: mapping distributional to model-theoretic semantic spaces. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 22–32. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339. Association for Computational Linguistics.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. *International Conference on Learning Representations*.

MichaelI. Jordan, Zoubin Ghahramani, TommiS. Jaakkola, and LawrenceK. Saul. 1999. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *International Conference on Learning Representations*.

Ponnambalam Kumaraswamy. 1980. A generalized probability density function for double-bounded random processes. *Journal of Hydrology*, 46(1-2):79–88.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117. Association for Computational Linguistics.

Zachary Chase Lipton. 2016. The mythos of model interpretability. *ICML Workshop on Human Interpretability in Machine Learning (WHI 2016)*.

Christos Louizos, Max Welling, and Diederik P Kingma. 2017. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continous relaxation of discrete random variables. *International Conference on Learning Representations*.

Andre Martins and Ramon Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International Conference on Machine Learning*, pages 1614–1623.

Julian McAuley, Jure Leskovec, and Dan Jurafsky. 2012. Learning attitudes and attributes from multi-aspect reviews. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 1020–1025. IEEE.

Eric Nalisnick and Padhraic Smyth. 2016. Stick-breaking variational autoencoders. *arXiv preprint arXiv:1605.06197*.

Vlad Niculae and Mathieu Blondel. 2017. A regularized framework for sparse and structured neural attention. In *Advances in Neural Information Processing Systems*, pages 3338–3348.

Vlad Niculae, André F. T. Martins, and Claire Cardie. 2018. Towards dynamic computation graphs via sparse latent structure. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 905–911. Association for Computational Linguistics.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255. Association for Computational Linguistics.

Hao Peng, Sam Thomson, and Noah A. Smith. 2018. Backpropagating through structured argmax using a spigot. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1863–1873. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286, Bejing, China. PMLR.

Marco Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 97–101. Association for Computational Linguistics.

Jason Tyler Rolfe. 2017. Discrete variational autoencoders. In *ICLR*.

Nicholas D. Socci, Daniel D. Lee, and H. Sebastian Seung. 1998. The rectified gaussian distribution. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 350–356. MIT Press.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642. Association for Computational Linguistics.

Sebastian Thrun. 1995. Extracting rules from artificial neural networks with distributed representations. In *Advances in neural information processing systems*, pages 505–512.

Ivan Titov and Ryan McDonald. 2008. A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of ACL*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

John Winn and Christopher M Bishop. 2005. Variational message passing. *Journal of Machine Learning Research*, 6(Apr):661–694.

Omar Zaidan and Jason Eisner. 2008. Modeling annotators: A generative approach to learning from annotator rationales. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 31–40, Honolulu, Hawaii. Association for Computational Linguistics.

Omar Zaidan, Jason Eisner, and Christine Piatko. 2007. Using "annotator rationales" to improve machine learning for text categorization. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 260–267. Association for Computational Linguistics.

Ye Zhang, Iain Marshall, and Byron C. Wallace. 2016. Rationale-augmented convolutional neural networks for text classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 795–804, Austin, Texas. Association for Computational Linguistics.

A Kumaraswamy distribution

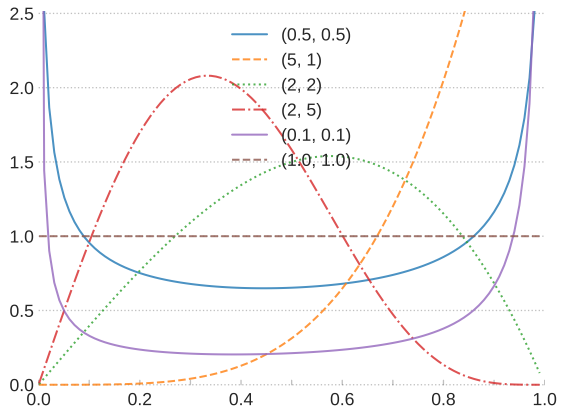


Figure 7: Kuma plots for various (a, b) parameters.

A Kumaraswamy-distributed variable $K \sim \mathrm{Kuma}(a, b)$ takes on values in the open interval $(0, 1)$ and has density

$$f_K(k; a, b) = abk^{a-1}(1 - k^a)^{b-1}\,, \tag{17}$$

where $a \in \mathbb{R}_{>0}$ and $b \in \mathbb{R}_{>0}$ are shape parameters. Its cumulative distribution takes a simple closed-form expression

$$\begin{aligned} F_K(k; a, b) &= \int_0^k f_K(\xi|a, b)\mathrm{d}\xi && \text{(18a)} \\ &= 1 - (1 - k^a)^b\,, && \text{(18b)} \end{aligned}$$

with inverse

$$F_K^{-1}(u; a, b) = \left(1 - (1 - u)^{1/b}\right)^{1/a}\,. \tag{19}$$

A.1 Generalised-support Kumaraswamy

We can generalise the support of a Kumaraswamy variable by specifying two constants $l < r$ and transforming a random variable $K \sim \mathrm{Kuma}(a, b)$ to obtain $T \sim \mathrm{Kuma}(a, b, l, r)$ as shown in (20, left).

$$t = l + (r - l)k \qquad k = {}^{(t-l)}/_{(r-l)} \tag{20}$$

The density of the resulting variable is

$$\begin{aligned} & f_T(t; a, b, l, r) && \text{(21a)} \\ &= f_K\left(\tfrac{t-l}{r-l}; a, b\right)\left|\frac{\mathrm{d}k}{\mathrm{d}t}\right| && \text{(21b)} \\ &= f_K\left(\tfrac{t-l}{r-l}; a, b\right)\frac{1}{(r-l)} && \text{(21c)} \end{aligned}$$

where $r - l > 0$ by definition. This *affine* transformation leaves the cdf unchanged, i.e.

$$\begin{aligned} F_T(t_0; a, b, l, r) &= \int_{-\infty}^{t_0} f_T(t; a, b, l, r)\mathrm{d}t \\ &= \int_{-\infty}^{t_0} f_K\left(\tfrac{t-l}{r-l}; a, b\right)\frac{1}{(r-l)}\mathrm{d}t \\ &= \int_{-\infty}^{\frac{t_0-l}{r-l}} f_K(k; a, b)\frac{1}{(r-l)}(r-l)\mathrm{d}k \\ &= F_K\left(\tfrac{t_0-l}{r-l}; a, b\right)\,. \end{aligned} \tag{22}$$

Thus we can obtain samples from this generalised-support Kumaraswamy by sampling from a uniform distribution $\mathcal{U}(0, 1)$, applying the inverse transform (19), then shifting and scaling the sample according to (20, left).

A.2 Rectified Kumaraswamy

First, we stretch a Kumaraswamy distribution to include 0 and 1 in its support, that is, with $l < 0$ and $r > 1$, we define $T \sim \mathrm{Kuma}(a, b, l, r)$. Then we apply a *hard-sigmoid* transformation to this variable, that is, $h = \min(0, \max(1, t))$, which results in a *rectified* distribution which gives support to the closed interval $[0, 1]$. We denote this rectified variable by

$$H \sim \mathrm{HardKuma}(a, b, l, r) \tag{23}$$

whose distribution function is

$$\begin{aligned} & f_H(h; a, b, l, r) = \\ & \mathbb{P}(h = 0)\delta(h) + \mathbb{P}(h = 1)\delta(h - 1) \\ & + \mathbb{P}(0 < h < 1)\frac{f_T(h; a, b, l, r)\mathbb{1}_{(0,1)}(h)}{\mathbb{P}(0 < h < 1)} \end{aligned} \tag{24}$$

where

$$\begin{aligned} & \mathbb{P}(h = 0) = \mathbb{P}(t \le 0) \\ & = F_T(0; a, b, l, r) = F_K({}^{-l}/_{(r-l)}; a, b) \end{aligned} \tag{25}$$

is the probability of sampling exactly 0, where

$$\begin{aligned} \mathbb{P}(h = 1) &= \mathbb{P}(t \ge 1) = 1 - \mathbb{P}(t < 1) \\ &= 1 - F_T(1; a, b, l, r) \\ &= 1 - F_K({}^{(1-l)}/_{(r-l)}; a, b) \end{aligned} \tag{26}$$

is the probability of sampling exactly 1, and

$$\mathbb{P}(0 < h < 1) = 1 - \mathbb{P}(h = 0) - \mathbb{P}(h = 1) \tag{27}$$

is the probability of drawing a continuous value in $(0, 1)$. Note that we used the result in (22) to express these probabilities in terms of the tractable cdf of the original Kumaraswamy variable.

A.3 Reparameterized gradients

Let us consider the case where we need derivatives of a function $\mathcal{L}(u)$ of the underlying uniform variable u, as when we compute reparameterized gradients in variational inference. Note that

$$\frac{\partial \mathcal{L}}{\partial u} = \frac{\partial \mathcal{L}}{\partial h} \times \frac{\partial h}{\partial t} \times \frac{\partial t}{\partial k} \times \frac{\partial k}{\partial u}\,, \qquad (28)$$

by chain rule. The term $\frac{\partial \mathcal{L}}{\partial h}$ depends on a differentiable observation model and poses no challenge; the term $\frac{\partial h}{\partial t}$ is the derivative of the hard-sigmoid function, which is 0 for $t < 0$ or $t > 1$, 1 for $0 < t < 1$, and undefined for $t \in \{0, 1\}$; the term $\frac{\partial t}{\partial k} = r - l$ follows directly from (20, left); the term $\frac{\partial k}{\partial u} = \frac{\partial}{\partial u} F_K^{-1}(u; a, b)$ depends on the Kumaraswamy inverse cdf (19) and also poses no challenge. Thus the only two discontinuities happen for $t \in \{0, 1\}$, which is a 0 measure set under the stretched Kumaraswamy: we say this reparameterisation is differentiable *almost everywhere*, a useful property which essentially circumvents the discontinuity points of the rectifier.

A.4 HardKumaraswamy PDF and CDF

Figure 8 plots the pdf of the HardKumaraswamy for various a and b parameters. Figure 9 does the same but with the cdf.

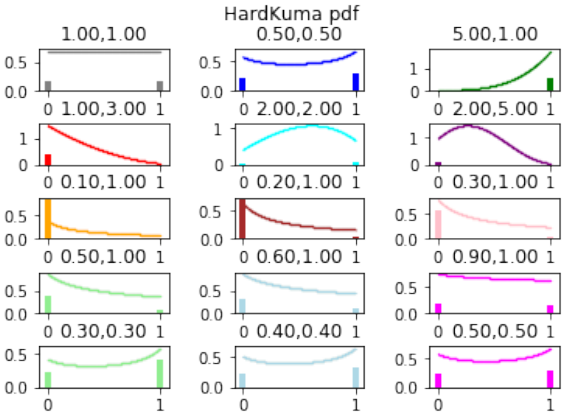


Figure 8: HardKuma pdf for various (a, b).

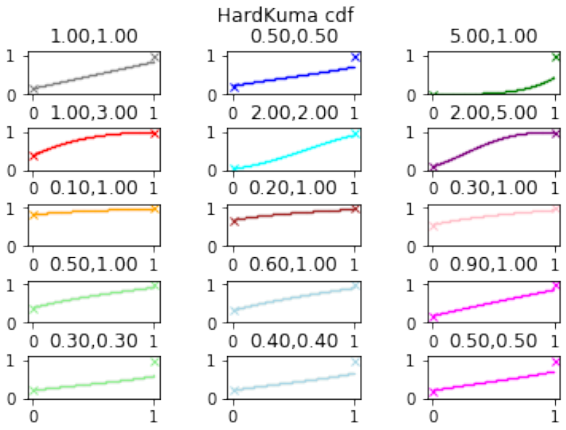


Figure 9: HardKuma cdf for various (a, b).

B Implementation Details

B.1 Multi-aspect Sentiment Analysis

Our hyperparameters are taken from Lei et al. (2016) and listed in Table 4. The pre-trained word embeddings and data sets are available online at `http://people.csail.mit.edu/taolei/beer/`. We train for 100 epochs and select the best models based on validation loss. For the MSE trade-off experiments on all aspects combined, we train for a maximum of 50 epochs.

Optimizer	Adam
Learning rate	0.0004
Word embeddings	200D (Wiki, fixed)
Hidden size	200
Batch size	256
Dropout	0.1, 0.2
Weight decay	$1 * 10^{-6}$
Cell	RCNN

Table 4: Beer hyperparameters.

For the Bernoulli baselines we vary L_0 weight λ_1 among $\{0.0002, 0.0003, 0.0004\}$, just as in the original paper. We set the fused lasso (coherence) weight λ_2 to $2 * \lambda_1$.

For the HardKuma models we set a target selection rate to the values targeted in Table 2, and optimize to this end using the Lagrange multiplier. We chose the fused lasso weight from $\{0.0001, 0.0002, 0.0003, 0.0004\}$.

B.1.1 Recurrent Unit

In our multi-aspect sentiment analysis experiments we use the RCNN of Lei et al. (2016). Intuitively, the RCNN is supposed to capture n-gram features that are not necessarily consecutive. We use the bigram version (filter width $n = 2$) used in

Lei et al. (2016), which is defined as:

$$\lambda_t = \sigma(W^\lambda \mathbf{x}_t + U^\lambda \mathbf{h}_{t-1} + \mathbf{b}^\lambda)$$
$$\mathbf{c}_t^{(1)} = \lambda_t \odot \mathbf{c}_{t-1}^{(1)} + (1 - \lambda_t) \odot W_1 \mathbf{x}_t$$
$$\mathbf{c}_t^{(2)} = \lambda_t \odot \mathbf{c}_{t-1}^{(2)} + (1 - \lambda_t) \odot (\mathbf{c}_{t-1}^{(1)} + W_2 \mathbf{x}_t)$$
$$\mathbf{h}_t = \tanh\Big(\mathbf{c}_t^{(2)} + \mathbf{b}\Big)$$

B.1.2 Expected values for dependent latent variables

The expected L_0 is a chain of nested expectations, and we solve each term

$$\begin{aligned} &\mathbb{E}_{p(z_i|x,z_{<i})}\left[\mathbb{I}[z_i \neq 0] \mid z_{<i}\right] \\ &= 1 - F_K\left(\tfrac{-l}{r-l}; a_i, b_i\right) \end{aligned} \tag{29}$$

as a function of a sampled prefix, and the shape parameters $a_i, b_i = g_i(x, z_{<i}; \phi)$ are predicted in sequence.

B.2 Sentiment Classification (SST)

For sentiment classification we make use of the PyTorch bidirectional LSTM module for encoding sentences, for both the rationale extractor and the classifier. The BiLSTM final states are concatenated, after which a linear layer followed by a softmax produces the prediction. Hyperparameters are listed in Table 5. We apply dropout to the embeddings and to the input of the output layer.

Optimizer	Adam
Learning rate	0.0002
Word embeddings	300D Glove (fixed)
Hidden size	150
Batch size	25
Dropout	0.5
Weight decay	$1 * 10^{-6}$
Cell	LSTM

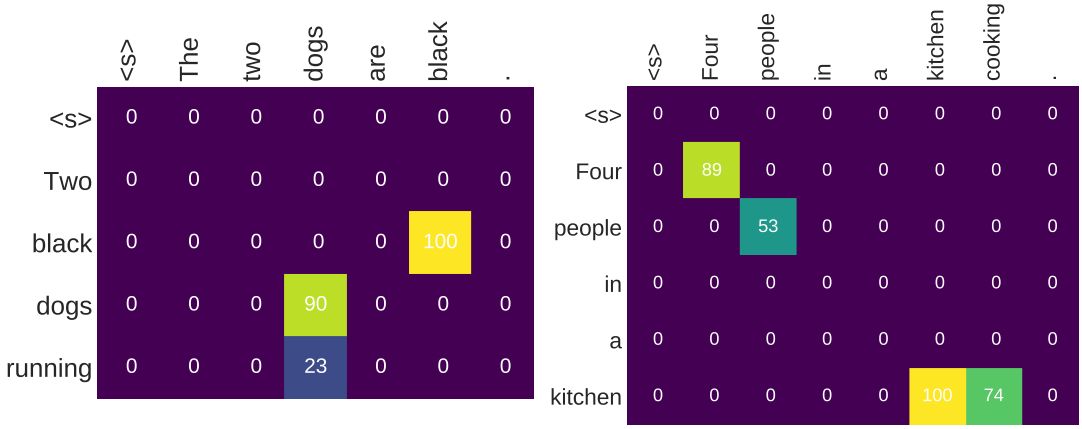
Table 5: SST hyperparameters.

B.3 Natural Language Inference (SNLI)

Our hyperparameters are taken from Parikh et al. (2016) and listed in Table 6. Different from Parikh et al. is that we use Adam as the optimizer and a batch size of 64. Word embeddings are projected to 200 dimensions with a trained linear layer. Unknown words are mapped to 100 unknown word classes based on the MD5 hash function, just as in Parikh et al. (2016), and unknown word vectors are randomly initialized. We train for 100 epochs, evaluate every 1000 updates, and select the best model based on validation loss. Figure 10 shows a correct and incorrect example with HardKuma attention for each relation type (entailment, contradiction, neutral).

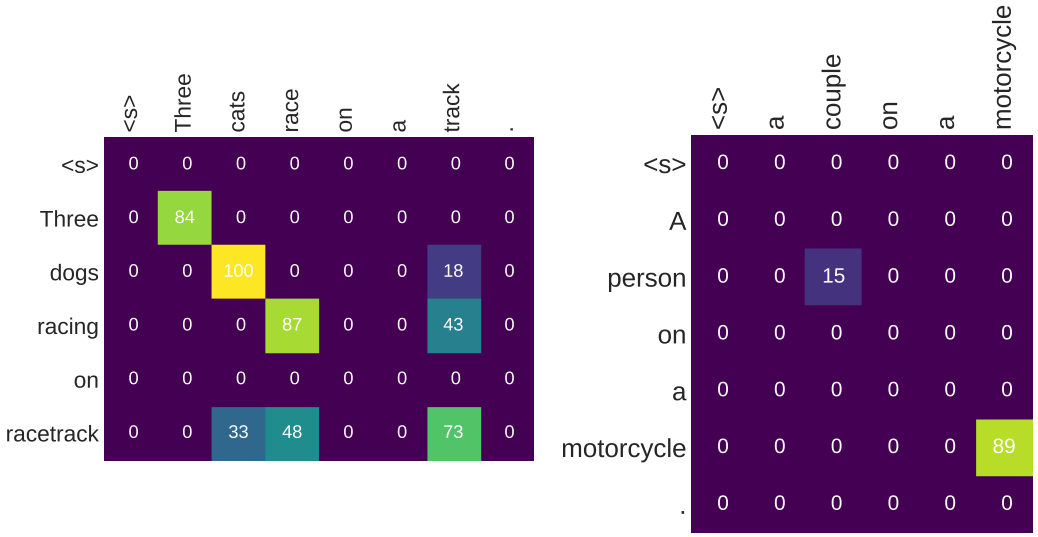
Optimizer	Adam
Learning rate	0.0001
Word embeddings	300D (Glove, fixed)
Hidden size	200
Batch size	64
Dropout	0.2

Table 6: SNLI hyperparameters.



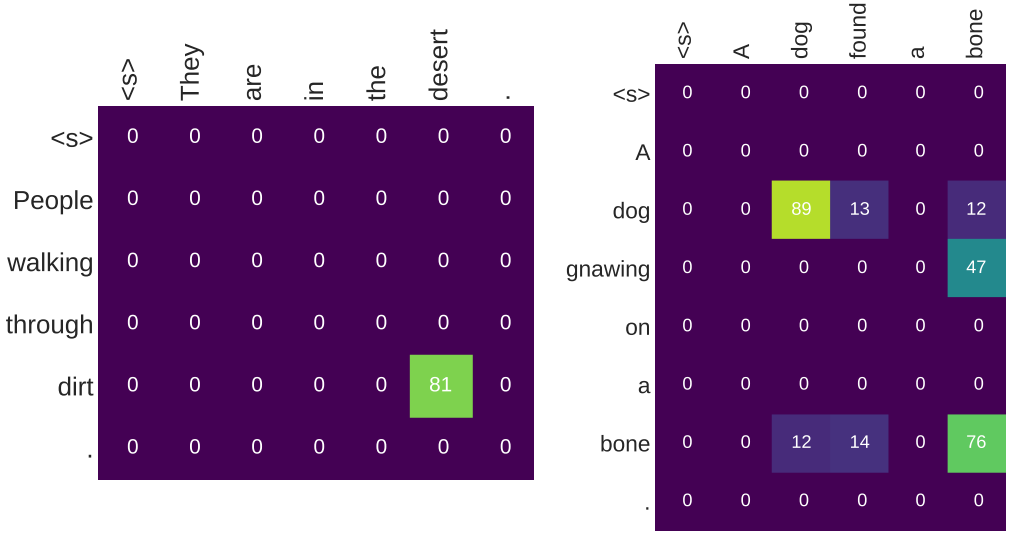
(a) Entailment (correct)

(b) Entailment (incorrect, pred: neutral)



(c) Contradiction (correct)

(d) Contradiction (incorrect, pred: entailment)



(e) Neutral (correct)

(f) Neutral (incorrect, pred: entailment)

Figure 10: HardKuma attention in SNLI for entailment, contradiction, and neutral.