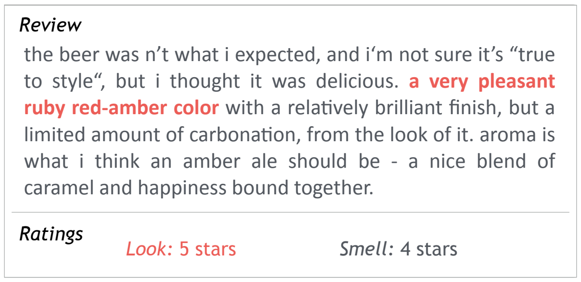
Rationalizing Neural Predictions

Tao Lei, Regina Barzilay and Tommi Jaakkola
MIT CSAIL
{taolei, regina, tommi}@csail.mit.edu

Abstract

Prediction without justification has limited applicability. As a remedy, we learn to extract pieces of input text as justifications – rationales – that are tailored to be short and coherent, yet sufficient for making the same prediction. Our approach combines two modular components, generator and encoder, which are trained to operate well together. The generator specifies a distribution over text fragments as candidate rationales and these are passed through the encoder for prediction. Rationales are never given during training. Instead, the model is regularized by desiderata for rationales. We evaluate the approach on multi-aspect sentiment analysis against manually annotated test cases. Our approach outperforms attention-based baseline by a significant margin. We also successfully illustrate the method on the question retrieval task.[1]



Review

the beer was n't what i expected, and i'm not sure it's "true to style", but i thought it was delicious. **a very pleasant ruby red-amber color** with a relatively brilliant finish, but a limited amount of carbonation, from the look of it. aroma is what i think an amber ale should be - a nice blend of caramel and happiness bound together.

Ratings *Look:* 5 stars *Smell:* 4 stars

Figure 1: An example of a review with ranking in two categories. The rationale for Look prediction is shown in bold.

1 Introduction

Many recent advances in NLP problems have come from formulating and training expressive and elaborate neural models. This includes models for sentiment classification, parsing, and machine translation among many others. The gains in accuracy have, however, come at the cost of interpretability since complex neural models offer little transparency concerning their inner workings. In many applications, such as medicine, predictions are used to drive critical decisions, including treatment options. It is necessary in such cases to be able to verify and understand the underlying basis for the decisions. Ideally, complex neural models would not only yield improved performance but would also offer interpretable justifications – rationales – for their predictions.

In this paper, we propose a novel approach to incorporating rationale generation as an integral part of the overall learning problem. We limit ourselves to extractive (as opposed to abstractive) rationales. From this perspective, our rationales are simply subsets of the words from the input text that satisfy two key properties. First, the selected words represent short and coherent pieces of text (e.g., phrases) and, second, the selected words must alone suffice for prediction as a substitute of the original text. More concretely, consider the task of multi-aspect sentiment analysis. Figure 1 illustrates a product review along with user rating in terms of two categories or aspects. If the model in this case predicts five star rating for color, it should also identify the phrase *"a very pleasant ruby red-amber color"* as the rationale underlying this decision.

In most practical applications, rationale genera-

[1]Code will be made available soon.

tion must be learned entirely in an unsupervised manner. We therefore assume that our model with rationales is trained on the same data as the original neural models, without access to additional rationale annotations. In other words, target rationales are never provided during training; the intermediate step of rationale generation is guided only by the two desiderata discussed above. Our model is composed of two modular components that we call the generator and the encoder. Our generator specifies a distribution over possible rationales (extracted text) and the encoder maps any such text to task specific target values. They are trained jointly to minimize a cost function that favors short, concise rationales while enforcing that the rationales alone suffice for accurate prediction.

The notion of what counts as a rationale may be ambiguous in some contexts and the task of selecting rationales may therefore be challenging to evaluate. We focus on two domains where ambiguity is minimal (or can be minimized). The first scenario concerns with multi-aspect sentiment analysis exemplified by the beer review corpus (McAuley et al., 2012). A smaller test set in this corpus identifies, for each aspect, the sentence(s) that relate to this aspect. We can therefore directly evaluate our predictions on the sentence level with the caveat that our the model makes selections on a finer level, in terms of words, not complete sentences. The second scenario concerns with the problem of retrieving similar questions. The extracted rationales should capture the main purpose of the questions. We can therefore evaluate the quality of rationales as a compressed proxy for the full text in terms of retrieval performance. Our model achieves high performance on both tasks. For instance, on the sentiment prediction task, our model achieves extraction accuracy of 96%, as compared to 38% and 81% obtained by the bigram SVM and a neural attention baseline.

2 Related Work

Developing sparse interpretable models holds considerably interest in the broader research community(Letham et al., 2015; Kim et al., 2015). The need for interpretability is even more pronounced with recent neural models. Efforts in this area include analyzing and visualizing state activation (Hermans and Schrauwen, 2013; Karpathy et al., 2015; Li et al., 2016), learning sparse interpretable word vectors (Faruqui et al., 2015b), and linking word vectors to semantic lexicons or word properties (Faruqui et al., 2015a; Herbelot and Vecchi, 2015).

Attention based models offer another lens to the inner workings of neural models (Bahdanau et al., 2015; Cheng et al., 2016; Martins and Astudillo, 2016; Chen et al., 2015; Xu and Saenko, 2015; Yang et al., 2015). Such models have been successfully applied to many NLP problems, improving both prediction accuracy as well as visualization and interpretability (Rush et al., 2015; Rocktäschel et al., 2016; Hermann et al., 2015).

Our work differs from past approaches in terms of what is meant by interpretable models (generating concise yet sufficient rationales) and how interpretation is derived (rationale generation). We explicitly aim to identify salient portions of the input text to justify predictions. Moreover, architecturally, we detach the rationale generation from how it is used (encoding) so as to be able to directly control types of rationales that are acceptable, and to facilitate broader modular use in other applications.

Finally, we contrast our work with rationale-based classification (Zaidan et al., 2007) which seeks to reduce supervised annotations by relying on richer annotations in the form of human-provided rationales. In our work, rationales are never given during training, and the goal is to learn to generate them.

3 Extractive Rationale Generation

We formalize here the task of extractive rationale generation and illustrate it in the context of neural models. To this end, consider a typical NLP task where we are provided with a sequence of words as input, namely $\mathbf{x} = \{x_1, \cdots, x_l\}$, where each $x_t \in \mathbb{R}^d$ denotes the vector representation of the i-th word. The learning problem is to map the input sequence $\mathbf{x}$ to a target vector in $\mathbb{R}^m$. For example, in multi-aspect sentiment analysis each coordinate of the target vector represents the response or rating pertaining to the associated aspect. In text retrieval, on the other hand, the target vectors are used to induce similarity assessments between input sequences. Broadly speaking, we can solve the associated learning problem by estimating a complex parameterized mapping $\mathbf{enc}(\mathbf{x})$ from input sequences to target vectors. We call this mapping an *encoder*. The training signal for these vectors is obtained either directly (e.g., multi-sentiment analysis) or via similarities (e.g., text retrieval). The challenge is that a complex neural encoder $\mathbf{enc}(\mathbf{x})$ reveals little about its internal workings and thus offers little in the way of justification for why a particular prediction was made.

In extractive rationale generation, our goal is to select a subset of the input sequence as a *rationale*. In order for the subset to qualify as a rationale it should satisfy two criteria: 1) the selected words should be interpretable and 2) they ought to suffice to reach nearly the same prediction (target vector) as the original input. In other words, a rationale must be short and sufficient. We will assume that a short selection is interpretable and focus on optimizing sufficiency under cardinality constraints.

We encapsulate the selection of words as a *rationale generator* which is another parameterized mapping $\mathbf{gen}(\mathbf{x})$ from input sequences to shorter sequences of words. Thus $\mathbf{gen}(\mathbf{x})$ must include only a few words and $\mathbf{enc}(\mathbf{gen}(\mathbf{x}))$ should result in nearly the same target vector as the original input passed through the encoder or $\mathbf{enc}(\mathbf{x})$. We can think of the generator as a tagging model where each word in the input receives a binary tag pertaining to whether it is selected to be included in the rationale. In our case, the generator is probabilistic and specifies a distribution over possible selections.

The rationale generation task is entirely unsupervised in the sense that we assume no explicit annotations about which words should be included in the rationale. Put another way, the rationale is introduced as a latent variable, a constraint that guides how to interpret the input sequence. The encoder and generator are trained jointly, in an end-to-end fashion so as to function well together.

4 Encoder and Generator

We use multi-aspect sentiment prediction as a guiding example to instantiate the two key components – the encoder and the generator. The framework itself generalizes to other tasks.

Encoder $\mathbf{enc}(\cdot)$: Given a training instance $(\mathbf{x}, \mathbf{y})$ where $\mathbf{x} = \{x_t\}_{t=1}^{l}$ is the input text sequence of length l and $\mathbf{y} \in [0,1]^m$ is the target m-dimensional sentiment vector, the neural encoder predicts $\tilde{\mathbf{y}} = \mathbf{enc}(\mathbf{x})$. If trained on its own, the encoder would aim to minimize the discrepancy between the predicted sentiment vector $\tilde{\mathbf{y}}$ and the gold target vector $\mathbf{y}$. We will use the squared error (i.e. L_2 distance) as the sentiment loss function,

$$\mathcal{L}(\mathbf{x}, \mathbf{y}) = \|\tilde{\mathbf{y}} - \mathbf{y}\|_2^2 = \|\mathbf{enc}(\mathbf{x}) - \mathbf{y}\|_2^2$$

The encoder could be realized in many ways such as a recurrent neural network. For example, let $\mathbf{h}_t = f_e(\mathbf{x}_t, \mathbf{h}_{t-1})$ denote a parameterized recurrent unit mapping input word $\mathbf{x}_t$ and previous state $\mathbf{h}_{t-1}$ to next state $\mathbf{h}_t$. The target vector is then generated on the basis of the final state reached by the recurrent unit after processing all the words in the input sequence. Specifically,

$$\mathbf{h}_t = f_e(\mathbf{x}_t, \mathbf{h}_{t-1}), \quad t = 1, \ldots, l$$
$$\tilde{\mathbf{y}} = \sigma_e(\mathbf{W}^e \mathbf{h}_l + \mathbf{b}^e)$$

Generator $\mathbf{gen}(\cdot)$: The rationale generator extracts a subset of text from the original input $\mathbf{x}$ to function as an interpretable summary. Thus the rationale for a given sequence $\mathbf{x}$ can be equivalently defined in terms of binary variables $\{\mathbf{z}_1, \cdots, \mathbf{z}_l\}$ where each $\mathbf{z}_t \in 0, 1$ indicates whether word $\mathbf{x}_t$ is selected or not. From here on, we will use $\mathbf{z}$ to specify the binary selections and thus $(\mathbf{z}, \mathbf{x})$ is the actual rationale generated (selections, input). We will use generator $\mathbf{gen}(\mathbf{x})$ as synonymous with a probability distribution over binary selections, i.e., $\mathbf{z} \sim \mathbf{gen}(\mathbf{x}) \equiv p(\mathbf{z}|\mathbf{x})$ where the length of $\mathbf{z}$ varies with the input $\mathbf{x}$.

In a simple generator, the probability that the t^{th} word is selected can be assumed to be conditionally independent from other selections given the input $\mathbf{x}$. That is, the joint probability $p(\mathbf{z}|\mathbf{x})$ factors according to

$$p(\mathbf{z}|\mathbf{x}) = \prod_{t=1}^{l} p(\mathbf{z}_t|\mathbf{x}) \quad \text{(independent selection)}$$

The component distributions $p(\mathbf{z}_t|\mathbf{x})$ can be modeled using a shared bi-directional recurrent neural network. Specifically, let $\overrightarrow{f}()$ and $\overleftarrow{f}()$ be the forward and backward recurrent unit, respectively, then

$$\begin{aligned}\overrightarrow{\mathbf{h}_t} &= \overrightarrow{f}(\mathbf{x}_t, \overrightarrow{\mathbf{h}_{t-1}}) \\ \overleftarrow{\mathbf{h}_t} &= \overleftarrow{f}(\mathbf{x}_t, \overleftarrow{\mathbf{h}_{t+1}}) \\ p(\mathbf{z}_t|\mathbf{x}) &= \sigma_z(\mathbf{W}^z[\overrightarrow{\mathbf{h}_t}; \overleftarrow{\mathbf{h}_t}] + \mathbf{b}^z)\end{aligned}$$

Independent but context dependent selection of words is often sufficient. However, the model is unable to select phrases or refrain from selecting the same word again if already chosen. To this end, we also introduce a dependent selection of words,

$$p(\mathbf{z}|\mathbf{x}) = \prod_{t=1}^{l} p(\mathbf{z}_t|\mathbf{x}, \mathbf{z}_1 \cdots \mathbf{z}_{t-1})$$

which can be also expressed as a recurrent neural network. To this end, we introduce another hidden state $\mathbf{s}_t$ whose role is to couple the selections. For example,

$$\begin{aligned}p(\mathbf{z}_t|\mathbf{x}, \mathbf{z}_{1,t-1}) &= \sigma_z(\mathbf{W}^z[\overrightarrow{\mathbf{h}_t}; \overleftarrow{\mathbf{h}_t}; \mathbf{s}_{t-1}] + \mathbf{b}^z) \\ \mathbf{s}_t &= f_z([\overrightarrow{\mathbf{h}_t}; \overleftarrow{\mathbf{h}_t}; \mathbf{z}_t], \mathbf{s}_{t-1})\end{aligned}$$

Joint objective: A rationale in our definition corresponds to the selected words, i.e., $\{\mathbf{x}_k|\mathbf{z}_k = 1\}$. We will use $(\mathbf{z}, \mathbf{x})$ as the shorthand for this rationale and, thus, $\mathbf{enc}(\mathbf{z}, \mathbf{x})$ refers to the target vector obtained by applying the encoder to the rationale as the input. Our goal here is to formalize how the rationale can be made short and meaningful yet function well in conjunction with the encoder. Our generator and encoder are learned jointly to interact well but they are treated as independent units for modularity.

The generator is guided in two ways during learning. First, the rationale that it produces must suffice as a replacement for the input text. In other words, the target vector (sentiment) arising from the rationale should be close to the gold sentiment. The corresponding loss function is given by

$$\mathcal{L}(\mathbf{z}, \mathbf{x}, \mathbf{y}) = \|\mathbf{enc}(\mathbf{z}, \mathbf{x}) - \mathbf{y}\|_2^2$$

Note that the loss function depends directly (parametrically) on the encoder but only indirectly on the generator via the sampled selection.

Second, we must guide the generator to realize short and coherent rationales. It should select only a few words and those selections should form phrases (consecutive words) rather than represent isolated, disconnected words. We therefore introduce an additional regularizer over the selections

$$\Omega(\mathbf{z}) = \lambda_1\|\mathbf{z}\| + \lambda_2 \sum_t |\mathbf{z}_t - \mathbf{z}_{t-1}|$$

where the first term penalizes the number of selections while the second one discourages transitions (encourages continuity of selections). Note that this regularizer also depends on the generator only indirectly via the selected rationale. This is because it is easier to assess the rationale once produced rather than directly guide how it is obtained.

Our final cost function is the combination of the two, $\text{cost}(\mathbf{z}, \mathbf{x}, \mathbf{y}) = \mathcal{L}(\mathbf{z}, \mathbf{x}, \mathbf{y}) + \Omega(\mathbf{z})$. Since the selections are not provided during training, we minimize the expected cost:

$$\min_{\theta_e, \theta_g} \sum_{(\mathbf{x}, \mathbf{y}) \in D} \mathbb{E}_{\mathbf{z} \sim \mathbf{gen}(\mathbf{x})} [\text{cost}(\mathbf{z}, \mathbf{x}, \mathbf{y})]$$

where θ_e and θ_g denote the set of parameters of the encoder and generator, respectively, and D is the collection of training instances. Our joint objective encourages the generator to compress the input text into coherent summaries that work well with the associated encoder it is trained with.

Minimizing the expected cost is challenging since it involves summing over all the possible choices of rationales $\mathbf{z}$. This summation could potentially be made feasible with additional restrictive assumptions about the generator and encoder. However, we assume only that it is possible to efficiently sample from the generator.

Doubly stochastic gradient We now derive a sampled approximation to the gradient of the expected cost objective. This sampled approximation is obtained separately for each input text $\mathbf{x}$ so as to work well with an overall stochastic gradient method. Consider therefore a training pair $(\mathbf{x}, \mathbf{y})$. For the parameters of the generator θ_g,

$$\begin{aligned}&\frac{\partial \mathbb{E}_{\mathbf{z} \sim \mathbf{gen}(\mathbf{x})} [\text{cost}(\mathbf{z}, \mathbf{x}, \mathbf{y})]}{\partial \theta_g} \\ &= \sum_{\mathbf{z}} \text{cost}(\mathbf{z}, \mathbf{x}, \mathbf{y}) \cdot \frac{\partial p(\mathbf{z}|\mathbf{x})}{\partial \theta_g} \\ &= \sum_{\mathbf{z}} \text{cost}(\mathbf{z}, \mathbf{x}, \mathbf{y}) \cdot \frac{\partial p(\mathbf{z}|\mathbf{x})}{\partial \theta_g} \cdot \frac{p(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})}\end{aligned}$$

Using the fact $(\log f(\theta))' = f'(\theta)/f(\theta)$, we get

$$\begin{aligned}
&\sum_{\mathbf{z}} \text{cost}(\mathbf{z}, \mathbf{x}, \mathbf{y}) \cdot \frac{\partial p(\mathbf{z}|\mathbf{x})}{\partial \theta_g} \cdot \frac{p(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})} \\
&= \sum_{\mathbf{z}} \text{cost}(\mathbf{z}, \mathbf{x}, \mathbf{y}) \cdot \frac{\partial \log p(\mathbf{z}|\mathbf{x})}{\partial \theta_g} \cdot p(\mathbf{z}|\mathbf{x}) \\
&= \mathbb{E}_{z \sim \mathbf{gen}(\mathbf{x})} \left[\text{cost}(\mathbf{z}, \mathbf{x}, \mathbf{y}) \frac{\partial \log p(\mathbf{z}|\mathbf{x})}{\partial \theta_g} \right]
\end{aligned}$$

The last term is the expected gradient where the expectation is taken with respect to the generator distribution over rationales $\mathbf{z}$. Therefore, we can simply sample a few rationales $\mathbf{z}$ from the generator $\mathbf{gen}(\mathbf{x})$ and use the resulting average gradient in an overall stochastic gradient method. A sampled approximation to the gradient with respect to the encoder parameters θ_e can be derived similarly,

$$\begin{aligned}
&\frac{\partial \mathbb{E}_{\mathbf{z} \sim \mathbf{gen}(\mathbf{x})} \left[\text{cost}(\mathbf{z}, \mathbf{x}, \mathbf{y}) \right]}{\partial \theta_e} \\
&= \sum_{\mathbf{z}} \frac{\partial \text{cost}(\mathbf{z}, \mathbf{x}, \mathbf{y})}{\partial \theta_e} \cdot p(\mathbf{z}|\mathbf{x}) \\
&= \mathbb{E}_{z \sim \mathbf{gen}(\mathbf{x})} \left[\frac{\partial \text{cost}(\mathbf{z}, \mathbf{x}, \mathbf{y})}{\partial \theta_e} \right]
\end{aligned}$$

Choice of recurrent unit We employ recurrent convolution (RCNN), a refinement of local-ngram based convolution. RCNN attempts to learn n-gram features that are not necessarily consecutive, and average features in a dynamic (recurrent) fashion. Specifically, for bigrams (filter width $n = 2$) RCNN computes $\mathbf{h}_t = f(\mathbf{x}_t, \mathbf{h}_{t-1})$ as follows

$$\begin{aligned}
\lambda_t &= \sigma(\mathbf{W}^\lambda \mathbf{x}_t + \mathbf{U}^\lambda \mathbf{h}_{t-1} + \mathbf{b}^\lambda) \\
\mathbf{c}_t^{(1)} &= \lambda_t \odot \mathbf{c}_{t-1}^{(1)} + (1 - \lambda_t) \odot (\mathbf{W}_1 \mathbf{x}_t) \\
\mathbf{c}_t^{(2)} &= \lambda_t \odot \mathbf{c}_{t-1}^{(2)} + (1 - \lambda_t) \odot (\mathbf{c}_{t-1}^{(1)} + \mathbf{W}_2 \mathbf{x}_t) \\
\mathbf{h}_t &= \tanh(\mathbf{c}_t^{(2)} + \mathbf{b})
\end{aligned}$$

RCNN has been shown to work remarkably in classification and retrieval applications (Lei et al., 2015a; Lei et al., 2015b) compared to other alternatives such CNNs and LSTMs. We use it for all the recurrent units introduced in our model.

5 Experiments

We evaluate the proposed joint model on two NLP applications: (1) multi-aspect sentiment analysis on product reviews and (2) similar text retrieval on AskUbuntu question answering forum.

Number of reviews	1580k
Avg length of review	144.9
Avg correlation between aspects	63.5%
Max correlation between two aspects	79.1%
Number of annotated reviews	994

Table 1: Statistics of the beer review dataset.

	D	d	l	$\|\theta\|$	MSE
SVM	260k	-	-	2.5M	0.0154
SVM	1580k	-	-	7.3M	0.0100
LSTM	260k	200	2	644k	0.0094
RCNN	260k	200	2	323k	**0.0087**

Table 2: Comparing neural encoders with bigram SVM model. MSE is the mean squared error on the test set. D is the amount of data used for training and development. d stands for the hidden dimension, l denotes the depth of network and $|\theta|$ denotes the number of parameters (i.e. number of features for SVM).

5.1 Multi-aspect Sentiment Analysis

Dataset We use the BeerAdvocate[2] review dataset used in prior work (McAuley et al., 2012).[3] This dataset contains 1.5 million reviews written by the website users. The reviews are naturally multi-aspect – each of the review contains multiple sentences describing the *overall* impression or one particular aspect of a beer, including *appearance*, *smell* (aroma), *palate* and the *taste*. In addition to the written text, the reviewer provides the ratings (on a scale of 0 to 5 stars) for each aspect as well as an overall rating. The ratings can be fractional (e.g. 3.5 stars), so we normalize the scores to $[0, 1]$ and use them as the (only) supervision for regression.

McAuley et al. (2012) also provided sentence-level annotations on around 1,000 reviews. Each sentence is annotated with one (or multiple) aspect label, indicating what aspect this sentence covers. We use this set as our test set to evaluate the precision of words in the extracted rationales.

Table 1 shows several statistics of the beer review dataset. The sentiment correlation between any pair of aspects (and the overall score) is quite high, getting 63.5% on average and a maximum of 79.1% (between the *taste* and *overall* score). If directly training the model on this set, the model can be confused due to such strong correlation. We therefore

[2]`www.beeradvocate.com`

[3]`http://snap.stanford.edu/data/web-BeerAdvocate.html`

Method	Appearance		Smell		Palate	
	% precision	% selected	% precision	% selected	% precision	% selected
SVM	38.3	13	21.6	7	24.9	7
Attention model	80.6	13	88.4	7	65.3	7
Generator (independent)	94.8	13	93.8	7	79.3	7
Generator (recurrent)	96.3	14	95.1	7	80.2	7

Table 3: Precision of selected rationales for the first three aspects. The precision is evaluated based on whether the selected words are in the sentences describing the target aspect, based on the sentence-level annotations. Best training epochs are selected based on the objective value on the development set (no sentence annotation is used).

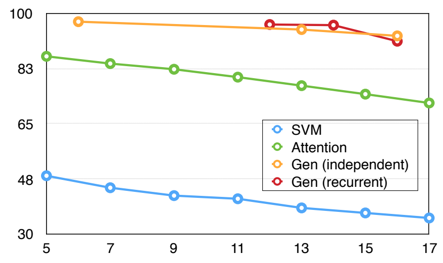


Figure 2: Precision (y-axis) when various percentages of text are extracted as rationales (x-axis) for the appearance aspect.

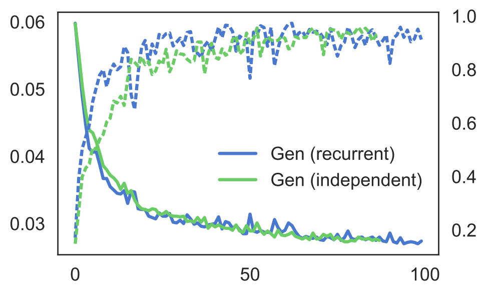


Figure 4: Learning curves of the optimized cost function on the development set and the precision of rationales on the test set. The smell (aroma) aspect is the target aspect.

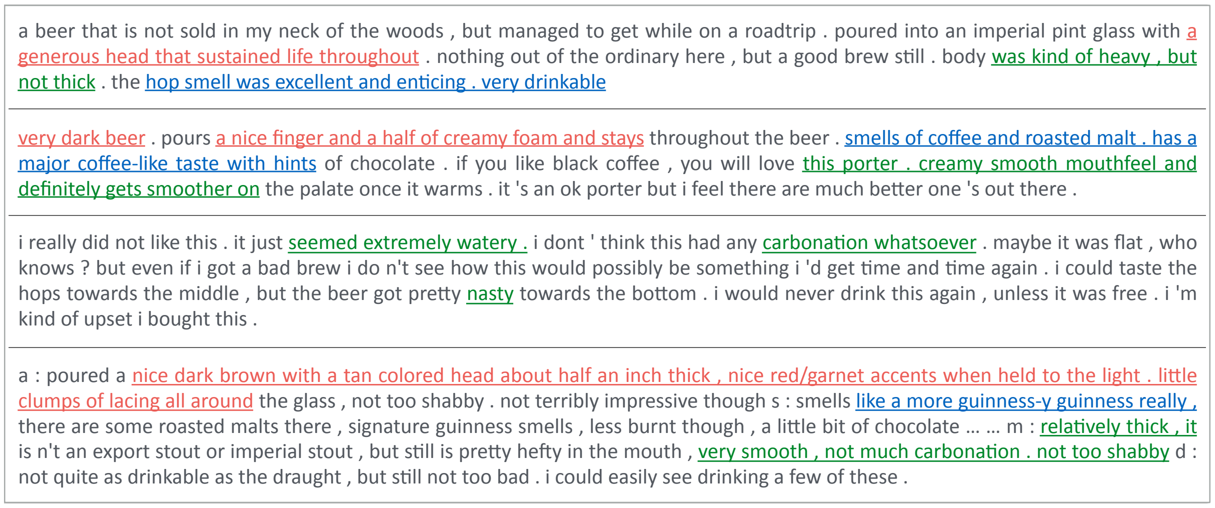
perform a preprocessing step, picking "less correlated" examples from the dataset.[4] This gives us a de-correlated subset for each aspect, each containing about 80k to 90k reviews. We use 10k as the development set. We focus on three aspects since the fourth aspect *taste* still gets $> 50\%$ correlation with the overall sentiment.

Encoder Performance Before training the joint model, it is worth assessing the neural encoder separately to check how accurate the neural network predicts the sentiment. To this end, we compare neural encoders with bigram SVM model, training medium and large SVM models using 260k and all 1580k reviews respectively. As shown in Table 2, the recurrent neural network models outperform the SVM model for sentiment prediction and also require less training data to achieve the performance. The LSTM and RCNN units obtain similar test error, getting 0.0094 and 0.0087 mean squared error respectively. The RCNN unit performs slightly better and uses less parameters. Based on the results, we choose the RCNN encoder network with 2 stacking layers and 200 hidden states in the joint model.

[4]Specifically, for each aspect we train a simple linear regression model to predict the rating of this aspect given the ratings of the other four aspects. We then keep picking reviews with largest prediction error until the sentiment correlation in the selected subset increases dramatically.

Rationale Selection To extract the supporting rationales for each aspect, we train the joint encoder-generator model on each de-correlated subset. Similar to the encoder, we use RCNN unit with 200 states as the forward and backward recurrent unit for the generator $\mathbf{gen}()$. The dependent selection generator has one additional recurrent layer. For this layer we use a small RCNN unit with 30 states, so the dependent version still has a number of parameters comparable to the independent version. The two versions of the generator have 358k and 323k parameters respectively. We set the cardinality regularization λ_1 between values $\{2e-4, 3e-4, 4e-4\}$ so the extracted rationale texts are neither too long nor too short. For simplicity, we set $\lambda_2 = 2\lambda_1$ to encourage local coherency of the extraction.

For comparison we use the bigram SVM model and implement an attention-based neural network model. The SVM model successively extracts unigram or bigram (from the test reviews) with the highest feature. The attention-based model learns a normalized attention vector of the input tokens (using similarly the forward and backward RNNs), then the model averages over the encoder states accordingly to the attention, and feed the averaged vector



a beer that is not sold in my neck of the woods , but managed to get while on a roadtrip . poured into an imperial pint glass with a generous head that sustained life throughout . nothing out of the ordinary here , but a good brew still . body was kind of heavy , but not thick . the hop smell was excellent and enticing . very drinkable

very dark beer . pours a nice finger and a half of creamy foam and stays throughout the beer . smells of coffee and roasted malt . has a major coffee-like taste with hints of chocolate . if you like black coffee , you will love this porter . creamy smooth mouthfeel and definitely gets smoother on the palate once it warms . it 's an ok porter but i feel there are much better one 's out there .

i really did not like this . it just seemed extremely watery . i dont ' think this had any carbonation whatsoever . maybe it was flat , who knows ? but even if i got a bad brew i do n't see how this would possibly be something i 'd get time and time again . i could taste the hops towards the middle , but the beer got pretty nasty towards the bottom . i would never drink this again , unless it was free . i 'm kind of upset i bought this .

a : poured a nice dark brown with a tan colored head about half an inch thick , nice red/garnet accents when held to the light . little clumps of lacing all around the glass , not too shabby . not terribly impressive though s : smells like a more guinness-y guinness really , there are some roasted malts there , signature guinness smells , less burnt though , a little bit of chocolate … … m : relatively thick , it is n't an export stout or imperial stout , but still is pretty hefty in the mouth , very smooth , not much carbonation . not too shabby d : not quite as drinkable as the draught , but still not too bad . i could easily see drinking a few of these .

Figure 3: Examples of extracted rationales indicating the sentiments of various aspects. The extracted texts for appearance, smell and plate are shown in red, blue and green color respectively. The last example is shortened for space.

to the output layer. Similar to the SVM model, the attention-based model can selects words based on their attention weights.

Table 3 presents the precision of the extracted rationales calculated based on sentence-level aspect annotations. The λ_1 regularization hyper-parameter is tuned so the two versions of our model extract similar number of words as rationales. The SVM and attention-based model are constrained similarly for comparison. Figure 2 further shows the precision when different amounts of text are extracted. For our model, this corresponds to changing the λ_1 regularization. As shown in the table and the figure, our encoder-generator networks extract text pieces describing the target aspect with high precision, ranging from 80% to 96% across the three aspects appearance, smell and palate. The SVM baseline performs poorly, achieving around 30% accuracy. The attention-based model achieves reasonable but worse performance than the rationale generator, suggesting the potential of directly modeling rationales as explicit extraction.

Figure 4 shows the learning curves of our model for the smell aspect. In the early training epochs, both the independent and recurrent dependent selection models fail to produce good rationales, getting low precision as a result. After a few epochs of exploration however, the models start to achieve high accuracy. We observe that the recurrent version learns more quickly in general, but both versions obtain close results in the end.

Finally we conduct a qualitative case study on the extracted rationales. Figure 3 presents several reviews, with highlighted rationales predicted by the model. Our rationale generator identifies key phrases or adjectives that indicate the sentiment of a particular aspect.

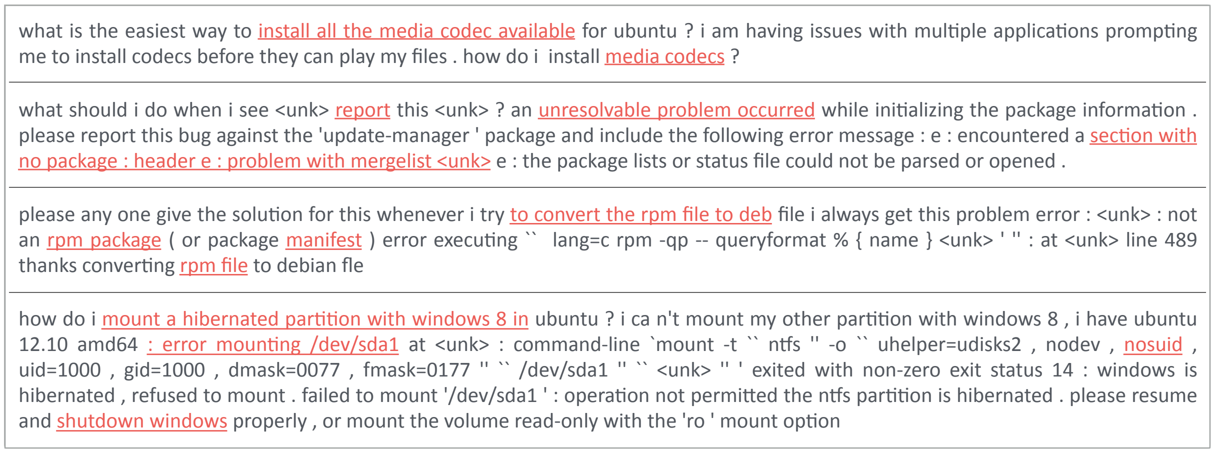
5.2 Similar Text Retrieval on QA Forum

Dataset For our second application, we use the real-world AskUbuntu[5] dataset used in recent work (dos Santos et al., 2015; Lei et al., 2015b). This set contains a set of 167k unique questions (each consisting a question title and a body) and 16k user-identified similar question pairs. Following previous work, this data is used to train the neural encoder that learns the vector representation of the input question, optimizing the cosine distance (i.e. cosine similarity) between similar questions against random non-similar ones. We use the "one-versus-all" hinge loss (i.e. positive versus other negatives) for the encoder, similar to (Lei et al., 2015b). During development and testing, the model is used to score 20 candidate questions given each query question, and a total of 400×20 query-candidate question pairs are annotated for evaluation[6].

Task/Evaluation Setup The question descriptions are often long and fraught with irrelevant details. In

[5] askubuntu.com

[6] https://github.com/taolei87/askubuntu


what is the easiest way to install all the media codec available for ubuntu ? i am having issues with multiple applications prompting me to install codecs before they can play my files . how do i install media codecs ?

what should i do when i see <unk> report this <unk> ? an unresolvable problem occurred while initializing the package information . please report this bug against the 'update-manager ' package and include the following error message : e : encountered a section with no package : header e : problem with mergelist <unk> e : the package lists or status file could not be parsed or opened .

please any one give the solution for this whenever i try to convert the rpm file to deb file i always get this problem error : <unk> : not an rpm package (or package manifest) error executing `` lang=c rpm -qp -- queryformat % { name } <unk> ' '' : at <unk> line 489 thanks converting rpm file to debian fle

how do i mount a hibernated partition with windows 8 in ubuntu ? i ca n't mount my other partition with windows 8 , i have ubuntu 12.10 amd64 : error mounting /dev/sda1 at <unk> : command-line `mount -t `` ntfs '' -o `` uhelper=udisks2 , nodev , nosuid , uid=1000 , gid=1000 , dmask=0077 , fmask=0177 '' `` /dev/sda1 '' `` <unk> '' ' exited with non-zero exit status 14 : windows is hibernated , refused to mount . failed to mount '/dev/sda1 ' : operation not permitted the ntfs partition is hibernated . please resume and shutdown windows properly , or mount the volume read-only with the 'ro ' mount option

Figure 5: Examples of extracted rationales of questions in the AskUbuntu domain.

	MAP (dev)	MAP (test)	%words
Full title	56.5	60.0	10.1
Full body	54.2	53.0	89.9
Independent	55.7	53.6	9.7
	56.3	52.6	19.7
Recurrent	56.1	54.6	11.6
	56.5	55.6	32.8

Table 4: Comparison between rationale models (middle and bottom rows) and the baselines using full title or body (top row).

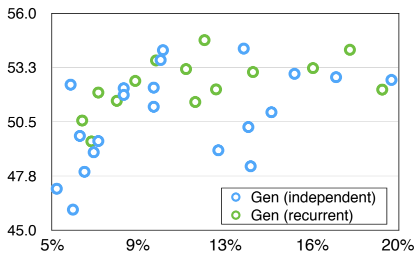


Figure 6: Retrieval MAP on the test set when various percentages of the texts are chosen as rationales. Data points correspond to models trained with different hyper-parameters.

this set-up, a fraction of the original question text should be sufficient to represent its content, and be used for retrieving similar questions. Therefore, we will evaluate rationales based on the accuracy of the question retrieval task, assuming that better rationales achieve higher performance. To put this performance in context, we also report the accuracy when full body of a question is used, as well as titles alone. The latter constitutes an upper bound on the model performance as in this dataset titles provide short, informative summaries of the question content. We evaluate the rationales using the mean average precision (MAP) of retrieval.

Results Table 4 presents the results of our rationale model. We explore a range of hyper-parameter values[7]. We include two runs for each version. The first one achieves the highest MAP on the development set, The second run is selected to compare the models when they use roughly 10% of question text (7 words on average). We also show the results of different runs in Figure 6. The rationales achieve the MAP up to 56.5%, getting close to using the titles. The models also outperform the baseline of using the noisy question bodies, indicating the the models' capacity of extracting short but important fragments.

Figure 5 shows the rationales for several questions in the AskUbuntu domain, using the recurrent version with around 10% extraction. Interestingly, the model does not always select words from the question title. The reasons are that the question body can contain the same or even complementary information useful for retrieval. Indeed, some rationale fragments shown in the figure are error messages, which are typically not in the titles but very useful to identify similar questions.

6 Conclusion

We propose a novel modular neural framework to automatically generate concise yet sufficient text fragments (i.e rationales) to justify predictions made by neural networks. The model can be trained in the end-to-end fashion, requiring no explicit annotation of rationales. We successfully illustrate the capacity of the model in two real-world applications.

[7] $\lambda_1 \in \{.008, .01, .012, .015\}$, $\lambda_2 = \{0, \lambda_1, 2\lambda_1\}$, dropout $\in \{0.1, 0.2\}$

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Kan Chen, Jiang Wang, Liang-Chieh Chen, Haoyuan Gao, Wei Xu, and Ram Nevatia. 2015. Abc-cnn: An attention based convolutional neural network for visual question answering. *arXiv preprint arXiv:1511.05960*.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

Cicero dos Santos, Luciano Barbosa, Dasha Bogdanova, and Bianca Zadrozny. 2015. Learning hybrid representations to retrieve semantically equivalent questions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 694–699, Beijing, China, July. Association for Computational Linguistics.

Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015a. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL*.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A. Smith. 2015b. Sparse overcomplete word vector representations. In *Proceedings of ACL*.

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*. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.

Michiel Hermans and Benjamin Schrauwen. 2013. Training and analysing deep recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 190–198.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*.

B Kim, JA Shah, and F Doshi-Velez. 2015. Mind the gap: A generative approach to interpretable feature selection and extraction. In *Advances in Neural Information Processing Systems*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2015a. Molding cnns for text: non-linear, non-consecutive convolutions. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1565–1575, Lisbon, Portugal, September. Association for Computational Linguistics.

Tao Lei, Hrishikesh Joshi, Regina Barzilay, Tommi Jaakkola, Katerina Tymoshenko, Alessandro Moschitti, and Lluis Marquez. 2015b. Semi-supervised question retrieval with gated convolutions. *arXiv preprint arXiv:1512.05726*.

Benjamin Letham, Cynthia Rudin, Tyler H. McCormick, and David Madigan. 2015. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *Annals of Applied Statistics*, 9(3):1350–1371.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in nlp. In *Proceedings of NAACL*.

André F. T. Martins and Ramón Fernandez Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. *CoRR*, abs/1602.02068.

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.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *International Conference on Learning Representations*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Huijuan Xu and Kate Saenko. 2015. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. *arXiv preprint arXiv:1511.05234*.

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. 2015. Stacked attention networks for image question answering. *arXiv preprint arXiv:1511.02274*.

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