Learning to Generate Product Reviews from Attributes

Li Dong[†], **Shaohan Huang**[‡], **Furu Wei**[‡], **Mirella Lapata**[†], **Ming Zhou**[‡] and **Ke Xu**[T]
[†]University of Edinburgh, Edinburgh, United Kingdom
[‡]Microsoft Research, Beijing, China
[T]Beihang University, Beijing, China
li.dong@ed.ac.uk, {fuwei, mingzhou}@microsoft.com,
buaahsh@gmail.com, mlap@inf.ed.ac.uk, kexu@nlsde.buaa.edu.cn

Abstract

Automatically generating product reviews is a meaningful, yet not well-studied task in sentiment analysis. Traditional natural language generation methods rely extensively on hand-crafted rules and predefined templates. This paper presents an attention-enhanced attribute-to-sequence model to generate product reviews for given attribute information, such as user, product, and rating. The attribute encoder learns to represent input attributes as vectors. Then, the sequence decoder generates reviews by conditioning its output on these vectors. We also introduce an attention mechanism to jointly generate reviews and align words with input attributes. The proposed model is trained end-to-end to maximize the likelihood of target product reviews given the attributes. We build a publicly available dataset for the review generation task by leveraging the Amazon book reviews and their metadata. Experiments on the dataset show that our approach outperforms baseline methods and the attention mechanism significantly improves the performance of our model.

1 Introduction

Nowadays, there are many popular online review sites (such as Amazon, and Yelp) that allow users to read and post reviews about books, electronics, restaurants, etc. The reviews are used to express opinions for different aspects of products, and have a wide variety of writing styles and different polarity strengths. As a result, much previous work has focused on how opinions are expressed in review data. For example, previous studies on sentiment analysis identify and extract subjective content in review data (Liu, 2015). However, few studies have explored building data-driven models that can generate product reviews for the given products and ratings, which is helpful to understand how a specific user comments for products. As shown in Figure 1, the input to our model is a set of attributes (such as user, product, and rating information), and our goal is to generate user- and product-specific reviews that agree with the input rating. These automatically generated reviews are useful for companies. For example, we could promote a product to users who have not bought it, by generating novel and personalized recommendations. We could also build a review writing assistant for E-commerce websites. After the website generates some candidate reviews according to the user's rating score, users could select one and refine it, which makes the procedure more user-friendly. Moreover, we can generate novel and personalized recommendations for every user, which makes the recommendation system more interpretable.

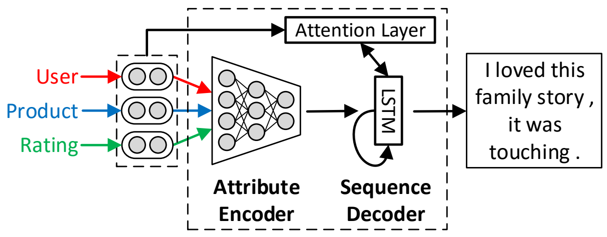

Figure 1: Our model learns to encode attributes into vectors, and then uses recurrent neural networks based on long short-term memory (LSTM) units to generate reviews by conditioning on the encoding vectors. An attention layer is used to learn soft alignments between attributes and generated words.

This attribute-conditioned review generation problem is very challenging due to the variety of candidate reviews that satisfy the input attributes. In other words, apart from the given attributes, there are other unknown or latent factors that influence the generated reviews, which renders the generation process non-deterministic. Moreover, although some attributes (such as rating) explicitly determine the usage of sentiment words, others (e.g., user information) implicitly influence word usage. So the model needs to handle both explicit and implicit clues. Additionally, the interactions between attributes are important to obtain the hidden factors used for generation. For example, different users tend to describe different aspects of a product and use different sentiment words to express a rating score.

In this paper, we propose a neural network based attribute-to-sequence model. As shown in Figure 1, our model contains three parts: attribute encoder, sequence decoder, and an attention mechanism. Specifically, we first use multilayer perceptrons to encode input attributes into vector representations that are used as latent factors for generating reviews. Next, the encoding vectors are fed into a coarse-to-fine sequence decoder. The decoder is built by stacking multiple layers of recurrent neural networks, which can generate words one by one conditioning on the encoding vectors. Besides, we introduce an attention layer into the proposed attribute-to-sequence model. The attention mechanism learns soft alignments between generated words and attributes, and adaptively computes encoder-side context vectors used to predict the next tokens. In order to evaluate our method, we build a dataset based on Amazon reviews and performed experiments on it. The experimental results show that the proposed model achieves superior performance against baseline methods. Moreover, we demonstrate that the attention mechanism significantly improves the performance of our model.

The contributions of this work are three-fold:

- We introduce the task of attribute-conditioned review generation, which is valuable for sentiment analysis, but not well studied previously.
- We propose an attention-enhanced attribute-to-sequence model in order to generate reviews conditioned on input attributes.
- We create a dataset based on Amazon book reviews and present empirical studies to show the proposed model outperforms several baseline methods.

2 Related Work

Sentiment analysis and opinion mining aim to identify and extract subjective content in text (Liu, 2015). Most previous work focuses on using rule-based methods or machine learning techniques for sentiment classification, which classifies reviews into different sentiment categories. Recently, deep learning has achieved promising results on sentiment analysis (Socher et al., 2011; Dong et al., 2014; Kim, 2014). Lipton et al. (2015) use character-level concatenated input recurrent neural networks as a generative model to predict rating and category for reviews. In contrast, our model is mainly evaluated on the review generation task rather than classification. Moreover, we use an attention mechanism in our encoder-decoder model, which has been proved very helpful in various tasks (Bahdanau et al., 2015; Xu et al., 2015), to generate user- and product-specific reviews. Maqsud (2015) compare latent Dirichlet allocation, Markov chains, and hidden Markov models for text generation on review data. However, we focus on generating product reviews conditioned on input attributes. Park et al. (2015) propose to retrieve relevant opinion sentences using product specifications as queries, while we work on generation instead of retrieval.

Our task definition is also related to concept-to-text generation (Konstas and Lapata, 2012; Konstas and Lapata, 2013), such as generating weather forecast or sportscasting from database records. A typical system contains three main stages: content planning, sentence planning, and surface realization. Mei et al. (2016) treat database records and output texts as sequences, and use recurrent neural networks to encode and decode them. In contrast, our input is a set of discrete attributes instead of database records or sequences. In addition, the contents of database records are strong constraints on results in concept-to-text generation. However, in our setting, user and product information implicitly indicates the style of generated reviews, which makes the results extremely diverse.

Another line of related work is the encoder-decoder model with neural networks. Specifically, an encoder is employed to encode input information into vectors, and then a decoder learns to predict results by conditioning outputs on the encoding vectors. This general framework is flexible because different neural networks can be used for encoders and decoders depending on the nature of inputs and outputs, which has been used to address various tasks. For example, recurrent neural networks are used to model sequences, such as machine translation (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014), syntactic parsing (Vinyals et al., 2015b), and semantic parsing (Dong and Lapata, 2016). Additionally, convolutional neural networks are employed for image data, such as image caption generation (Vinyals et al., 2015a), and video description generation (Donahue et al., 2015; Venugopalan et al., 2015). Our model employs multilayer perceptron to encode attribute information, and uses recurrent neural networks to decode product reviews. In order to better handle alignments between inputs and outputs, the attention mechanism is introduced for the encoder-decoder model. The attention model boosts performance for various tasks (Bahdanau et al., 2015; Luong et al., 2015; Xu et al., 2015). In our work, we use the attention mechanism to learn soft alignments between input attributes and output sequences, which has not, to our knowledge, been studied in previous work. Dosovitskiy et al. (2015) propose to use generative convolutional neural networks to generate images of chairs given chair type, viewpoint and color. Similarly, Yan et al. (2016) use variational auto-encoders to generate face images conditioned on visual attributes. However, our goal is to generate texts instead of images. Moreover, we learn a neural attention model to attend over input attributes during generation.

3 Modelling Approach

To begin with, we state the product review generation problem as follows. Given input attributes $a = \left(a_1, \cdots, a_{|a|}\right)$, our goal is to generate a product review $r = \left(y_1, \cdots, y_{|r|}\right)$ maximizing the conditional probability $p\left(r|a\right)$. Notice that number of attributes $|a|$ is fixed, while the review r is considered a word sequence of variable length. We use the user ID, product ID, and rating as attributes, so $|a|$ is set to 3 in our task. The training data are attributes paired with corresponding reviews. The model learns to compute the likelihood of generated reviews given input attributes. This conditional probability $p\left(r|a\right)$ is decomposed to:

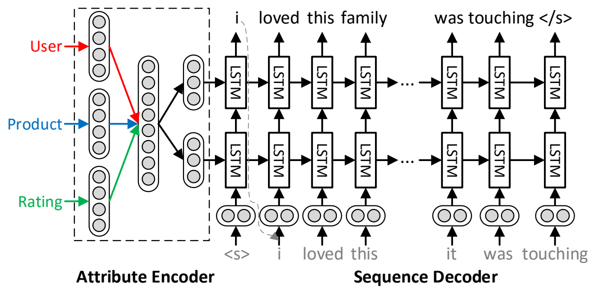

Figure 2: Attribute-to-sequence model without attention mechanism.

$$p\left(r|a\right) = \prod_{t=1}^{|r|} p\left(y_t|y_{<t}, a\right) \tag{1}$$

where $y_{<t} = \left(y_1, \cdots, y_{t-1}\right)$.

Our method consists of three parts, i.e., an attribute encoder, a sequence decoder, and an attention layer. The attribute encoder employs multilayer perceptrons to encode attributes a to vectors. To be specific, we represent the attributes as vectors. Next, the concatenation of these vectors is fed into a hidden layer to obtain the encoding vectors. After we obtain the encoding vectors, the sequence decoder stacks L-layer recurrent neural networks (RNNs) to generate reviews conditioning on these vectors. During decoding, RNNs recurrently compute n-dimensional hidden vectors which are used to predict output words for different time steps. In order to better utilize encoder-side information, an attention layer is introduced to learn soft alignments between attributes and output words. For every decoding time step, we use the current hidden vector to compute attention scores over attribute vectors. Then, a weighted sum of attribute vectors is used as the context vector to predict output words.

We first describe the attribute-to-sequence model without using neural attention in Section 3.1 and Section 3.2. Next, we introduce the attention mechanism in Section 3.3.

3.1 Attribute Encoder

We use multilayer perceptrons with one hidden layer to encode attribute information into a vector as shown in Figure 2. At first, input attributes $a = \left(a_1, \cdots, a_{|a|}\right)$ are represented by low-dimensional vectors. The attribute a_i's vector $\mathbf{g}\left(a_i\right)$ is computed via:

$$\mathbf{g}\left(a_i\right) = W_i^a \mathbf{e}\left(a_i\right) \tag{2}$$

where $W_i^a \in \mathbb{R}^{m \times |a_i|}$ is a parameter matrix, m is the dimension of embedding, and $\mathbf{e}(a_i) \in \{0,1\}^{|a_i|}$ is a one-hot vector representing the presence or absence of a_i. Then, these attribute vectors are concatenated and fed into a hidden layer which outputs the encoding vector. The output of the hidden layer is computed as:

$$\mathbf{a} = \tanh\left(H[\mathbf{g}(a_1), \cdots, \mathbf{g}\left(a_{|a|}\right)] + \mathbf{b}_a\right) \quad (3)$$

where $[\mathbf{g}(a_1), \cdots, \mathbf{g}\left(a_{|a|}\right)]$ are concatenated attribute vectors, $\tanh$ is a nonlinearity function, $H \in \mathbb{R}^{Ln \times |a|m}$ is a weight matrix, and $\mathbf{b}_a \in \mathbb{R}^{Ln}$ is the bias. Next, the vector $\mathbf{a}$ is used to initialize the n-dimensional hidden vectors of the L-layer recurrent neural networks in the decoder.

3.2 Sequence Decoder

As shown in Figure 2, the decoder is built upon multilayer recurrent neural networks (RNNs) with long short-term memory (LSTM) units. RNNs use vectors to represent information for the current time step and recurrently compute the next hidden states. In our work, we stack multiple layers of RNNs in our architecture. Additionally, a long short-term memory (Hochreiter and Schmidhuber, 1997) unit is employed to better handle long sequences. The LSTM introduces several gates and explicit memory cells to memorize or forget information, which enables networks learn more complicated patterns. Let $\mathbf{h}_t^l \in \mathbb{R}^n$ denote an n-dimensional hidden vector in layer l and time step t. $\mathbf{h}_t^l$ is computed via:

$$\mathbf{h}_t^l = f\left(\mathbf{h}_{t-1}^l, \mathbf{h}_t^{l-1}\right) \quad (4)$$

where $\mathbf{h}_t^0 = W^r \mathbf{e}(y_{t-1})$ is the word embedding of the previous predicted word, $W^r \in \mathbb{R}^{n \times |V_r|}$ is a parameter matrix, $|V_r|$ is the vocabulary size, and $\mathbf{e}(y_{t-1})$ is a one-hot vector used to extract word vector for y_{t-1}. We follow the architecture of LSTM unit described in Zaremba et al. (2015). To be specific, the unit is given by:

$$\begin{pmatrix} \mathbf{i} \\ \mathbf{f} \\ \mathbf{o} \\ \mathbf{g} \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} W^l \begin{pmatrix} \mathbf{h}_t^{l-1} \\ \mathbf{h}_{t-1}^l \end{pmatrix}$$
$$\mathbf{p}_t^l = \mathbf{f} \odot \mathbf{p}_{t-1}^l + \mathbf{i} \odot \mathbf{g} \quad (5)$$
$$\mathbf{h}_t^l = \mathbf{o} \odot \tanh\left(\mathbf{p}_t^l\right)$$

where $\tanh$, sigm, and $\odot$ are element-wise operators, and $W^l \in \mathbb{R}^{4n \times 2n}$ is a weight matrix for the l-th layer.

Once the input attributes are encoded to the vector $\mathbf{a} \in \mathbb{R}^{Ln}$ by Equation (3), the encoding vector is split into L vectors to initialize the hidden vectors of the first time step in decoder. Then, RNNs compute hidden vectors recurrently and predict output words using the hidden vectors of the topmost layer $\mathbf{h}_t^L$. For the vanilla model without using an attention mechanism, the predicted distribution of the t-th output word is:

$$p\left(y_t | y_{<t}, a\right) = \text{softmax}_{y_t}\left(W^p \mathbf{h}_t^L\right) \quad (6)$$

where $W^p \in \mathbb{R}^{|V_r| \times n}$ is a parameter matrix.

3.3 Attention Mechanism

The attention mechanism is introduced to better utilize encoder-side information. As indicated in Equation (6), the vanilla model does not directly use attribute vectors to generate sequences. Intuitively, the model can concentrate on different parts of encoding information to predict the next word. Previous work has proved this idea significantly improves performance especially for long sequences (Bahdanau et al., 2015; Vinyals et al., 2015b; Luong et al., 2015).

Figure 3 demonstrates how to compute the encoder-side context vector and use it to predict output words. For the t-th time step of the decoder, we compute the attention score of attribute a_i via:

$$s_i^t = \exp\left(\tanh\left(W^s\left[\mathbf{h}_t^L, \mathbf{g}(a_i)\right]\right)\right)/Z \quad (7)$$

where the brackets $[\cdot, \cdot]$ denote concatenation, Z is a normalization term that ensures $\sum_{i=1}^{|a|} s_i^t = 1$, and $W^s \in \mathbb{R}^{1 \times (n+m)}$ is a parameter matrix. Next, the attention context vector $\mathbf{c}^t$ is obtained by:

$$\mathbf{c}^t = \sum_{i=1}^{|a|} s_i^t \, \mathbf{g}(a_i) \quad (8)$$

which is a weighted sum of attribute vectors. We further employ the vector $\mathbf{c}^t$ to predict the t-th output token as:

$$\mathbf{h}_t^{att} = \tanh\left(W_1 \mathbf{c}^t + W_2 \mathbf{h}_t^L\right) \quad (9)$$

$$p\left(y_t | y_{<t}, a\right) = \text{softmax}_{y_t}\left(W^p \mathbf{h}_t^{att}\right) \quad (10)$$

where $W^p \in \mathbb{R}^{|V_r| \times n}$, $W_1 \in \mathbb{R}^{n \times m}$ and $W_2 \in \mathbb{R}^{n \times n}$ are three parameter matrices.

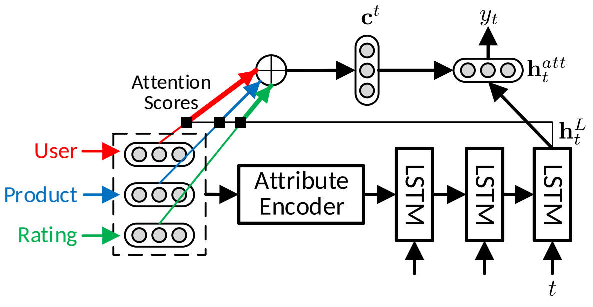

Figure 3: Attention scores are computed by attribute vectors and the current hidden vector of the decoder. Then, the encoder-side context vector is obtained in the form of a weighted sum, which is further used to predict the word distribution.

3.4 Model Training

We aim at maximizing the likelihood of generated reviews given input attributes for the training data. So we define the optimization problem as:

$$\text{maximize} \sum_{(a,r)\in\mathcal{D}} \log p\,(r|a) \tag{11}$$

where $\mathcal{D}$ is the dataset of all attribute-review training pairs, and $p\,(r|a)$ is defined as shown in Equation (1). In order to avoid overfitting, we insert dropout layers between different LSTM layers as suggested in Zaremba et al. (2015). The mini-batched RMSProp (Tieleman and Hinton, 2012) algorithm is used to optimize the objective function.

3.5 Inference

At test time, we first use the encoder to encode input attributes into vectors, and use them to initialize the LSTM units of the decoder. Then, the decoder predicts a review $\hat{r}$ that maximizes the conditional probability defined in Equation (1):

$$\hat{r} = \arg\max_{r'} p\left(r'|a\right) \tag{12}$$

where r' is a candidate review. Because we decompose this probability as shown in Equation (1), we can use beam search or greedy search to generate words, which avoids iterating over all candidate reviews. In order to determine the termination of the generation process, we add a special token $</s>$ to the end of every output review. The generation terminates once this token is emitted.

4 Experiments

We first introduce a new dataset for this task and compare our method with several baseline approaches. Then we conduct some ablation experiments and present model analysis to help us understand what the model learns.

4.1 Dataset Description

Our dataset is built upon Amazon product data (McAuley et al., 2015) that includes reviews and metadata spanning from May 1996 to July 2014 with duplicates removed. The products of the book domain are used in our experiments. Every review is paired with three attributes, i.e., user ID, product ID and rating. We filter books and users which do not occur at least 6 and 15 times, respectively. The reviews whose lengths are greater than 60 words are filtered. Because we observe that long reviews mainly describe the plots of books, while our goal is to generate reviews expressing opinions. The average review length is about 35 words, and the average number of sentences is 3. The dataset contains 937,033 reviews paired with attributes. Specifically, we have 80,256 books, 19,675 users, and 5 rating levels. The word vocabulary size is 161K. Then, the whole dataset is randomly split into TRAIN, DEV, and TEST (70%/10%/20%). The dataset is available at `https://goo.gl/TFjEH4`.

4.2 Settings

We used NLTK (Bird et al., 2009) to tokenize the reviews, and employed the Wikipedia list of common misspellings to correct misspelled words. We kept words that appeared more than 10 times in our vocabulary. The training hyperparameters are selected based on the results of the DEV set. The dimension of attribute vectors is set to 64. The dimensions of word embeddings and hidden vectors are set to 512 in the sequence decoder. Moreover, we stack two layers of recurrent neural networks with LSTM units to generate reviews. All the parameters are randomly initialized by sampling from a uniform distribution $[-0.08, 0.08]$. The batch size, smoothing constant and base learning rate of RMSProp are set to 50, 0.95 and 0.002, respectively. After 10 epochs, the learning rate is decreased by a factor of 0.97 at the end of every epoch as suggested in Karpathy et al. (2016). The dropout rate is set to 0.2 for regularization. We also clamp gradient values into the range $[-5, 5]$ to avoid the exploding gradient problem (Pascanu et al., 2013). The number of epochs is determined by early stopping on the DEV set. At test time,

Method	**BLEU-4** (%)	**BLEU-1** (%)
Rand	0.86	20.36
MELM	1.28	21.59
NN-pr	1.53	22.44
NN-ur	3.61	26.37
Att2Seq	4.51	30.24
Att2Seq+A	**5.03***	**30.48***

Table 1: Evaluation results on the TEST set of Amazon data. *: significantly better than the second best score ($p < 0.05$).

we use the greedy search algorithm to generate reviews.

4.3 Evaluation Results

The BLEU (Papineni et al., 2002) score is used for automatic evaluation, which has been shown to correlate well with human judgment on many generation tasks. The BLEU score measures the precision of n-gram matching by comparing the generated results with references, and penalizes length using a brevity penalty term. We compute BLEU-1 (unigram) and BLEU-4 (up to 4 grams) in experiments.

4.3.1 Comparison with Baseline Methods

We describe the comparison methods as follows:

Rand. The predicted results are randomly sampled from all the reviews in the TRAIN set. This baseline method suggests the expected lower bound for this task.

MELM. Maximum Entropy Language Model uses n-gram (up to trigram) features, and the feature template attribute&n-gram (up to bigram). The feature hashing technique is employed to reduce memory usage in each feature group. Noise contrastive estimation (Gutmann and Hyvrinen, 2010) is used to accelerate the training by dropping the normalization term, with 20 contrastive samples in training.

NN-pr. This Nearest Neighbor based method retrieves the reviews that have the same product ID and rating as the input attributes in the TRAIN set. Then we randomly choose a review from them, and use it as the prediction.

NN-ur. The same method as NN-pr but uses both user ID and rating to retrieve candidate reviews.

Att2Seq. Our attribute-to-sequence method described in Section 3. Notice that the attention model is not used.

Method	MELM	Att2Seq	Att2Seq+A
Accuracy (%)	59.00	88.67	**93.33***

Table 2: We manually annotate some polarity labels (positive or negative) for generated reviews and compute accuracy by comparing them with the input ratings. *: significantly better than the second best accuracy ($p < 0.05$).

Att2Seq+A. Our method with an attention mechanism.

As shown in Table 1, we compute BLEU scores for these methods. The results of random guess indicate that this task is non-trivial to obtain reasonable performance. MELM performs worse than nearest neighbor search due to the sparsity of lexicalized features, while our model employs distributed representations to avoid using sparse indicator features. Then, we evaluate the NN methods that use different attributes to retrieve reviews, which is a strong baseline for the generation task. The results show that our method outperforms the baseline methods. Moreover, the improvements of the attention mechanism are significant with $p < 0.05$ according to the bootstrap resampling test (Koehn, 2004). We further show some examples to analyze the attention model in Section 4.4.

4.3.2 Polarity of Generated Reviews

In order to evaluate whether the polarities of generated reviews correspond to their input ratings, we randomly sample some generated reviews and manually annotate their polarity labels. Specifically, we regard the rating 1-2 as negative and 4-5 as positive, and then evaluate performance by computing their classification accuracy. We randomly sample 150 negative examples and 150 positive examples for each method. Next, we ask two graduate students to classify the generated reviews to positive, negative, and indeterminable/neutral. About 93% of examples are annotated with the same labels by two annotators. Table 2 shows our method significantly outperforms others ($p < 0.05$). For Att2Seq+A, some generated reviews are classified to indeterminable/neutral because they contain mixed opinions towards different aspects of books.

4.3.3 Ablation Experiments

In order to evaluate the contributions of model components in our method, we compare to the variants of our model. These models are described

Method	BLEU-4 (%)	BLEU-1 (%)
Att2Seq+A	5.01	30.23
AvgEnc	4.07	28.13
NoStack	4.73	29.58
w/o user	4.10	26.87
w/o product	4.13	27.15
w/o rating	4.12	27.98

Table 3: Model ablation results on the DEV set.

as follows:

AvgEnc. This model uses the average of attribute vectors as the encoding vector, rather than multilayer perceptrons.

NoStack. The method only uses one-layer recurrent neural networks for the sequence decoder.

w/o user/product/rating. This variant does not use the corresponding attribute as input. These results indicate the importance of different information for our model.

As shown in Table 3, we compute BLEU-4 and BLEU-1 scores for our full model and the different variants on the DEV set. The ablation model AvgEnc performs worse than Att2Seq+A. This indicates that multilayer perceptrons can better handle interactions between attributes, outperforming simple averaging of input vectors. Next, we compare to the model without stacking multiple layers of recurrent neural networks as described in Section 3.2. The results demonstrate that deep architectures can improve generation performance. For the next group of variants, we find that removing user, product and rating information harms performance, which indicates that all three attributes contribute to generating relevant reviews.

4.4 The Attention Mechanism

As described in Section 3.3, the attention mechanism learns soft alignment scores between generated words and input attributes. These scores are used to obtain encoder-side context vectors that can better utilize attribute information to predict the next word.

Figure 4 shows three generated examples with different input ratings. The attention scores are represented by gray scales and are column-wisely normalized as described in Equation (7). Firstly, we explain the attention scores over rating information. The input rating of the first example is 1. We find that the phrases "*n't expecting much*", "*n't like*" and "*a little too much*" have larger attention scores on the rating attribute. This demonstrates rating information has more effect on generating these sentiment words. Next, we increase the rating score to 3 in the second example. The generated review expresses a mixed opinion for different aspects of the book. As indicated by the attention scores, we know that the sentiment words "*loved*", "*little slow*", and "*not like*" attend more to rating information. The last example is a positive review with a rating of 5. The attention scores demonstrate the phrases "*loved*", "*ca n't wait*", and "*hope * writes another book*" are used to express polarity. Similarly, the attention scores over user and product information indicate how the generated words are aligned with these two input attributes. For instance, the word "*characters*" has higher attention scores over the product attribute in the first and second example. This indicates that users tend to comment about the characters in this book's reviews.

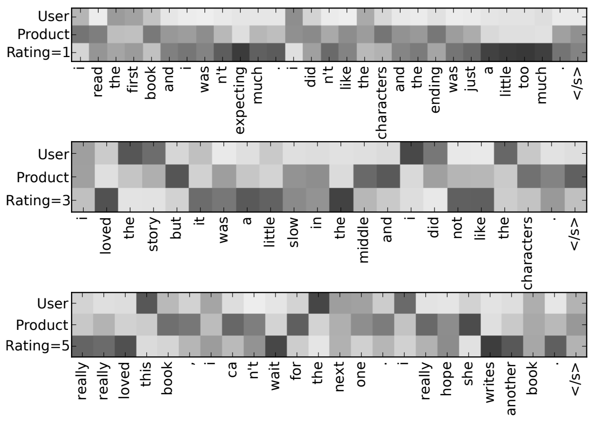

Figure 4: Examples of attention scores (Equation (7)) over three attributes. Darker color indicates higher attention score.

4.5 Generated Examples

As shown in Table 4, we sample products and users to generate some examples with different ratings. The special unknown token is removed from the vocabulary of the decoder in the generation process. We keep two attributes fixed and change the other one in every group to show the effects of input.

In the first group, we change the rating from 1 to 5 and keep the others unchanged. The results show that the polarity of generated reviews changes with the rating. For instance, the words "*nice*" and "*liked*" are used for the rating of 3, while the words "*very good*" and "*enjoyed*" are employed for the rating of 5. Moreover, both ex-

U	P	R	**Generated Review**
A	V	1	i'm sorry to say this was a very boring book. i didn't finish it. i'm not a new fan of the series, but this was a disappointment.
A	V	3	this was a nice story. i liked the characters and the story line. i'm not sure i'd read another by this author.
A	V	5	this was a very good book. i enjoyed the characters and the story line. i'm looking forward to reading more in this series.
B	W	5	i couldn't put it down. it was a great love story. i can't wait to read the next one.
C	W	5	enjoyable story that keeps you turning the pages. the characters are well developed and the plot is excellent. i would recommend this book to anyone who enjoys a good love story.
D	W	5	i loved this book. i could not put it down. i loved this story and the characters. i will be reading the next book.
E	X	1	i read this book because i was looking for something to read. this book was just too much like the others. i thought the author was going to be a good writer, but i was disappointed.
E	Y	1	i was disappointed. i read the first chapter and then i was bored. i read the whole thing, but i just couldn't get into it.
E	Z	1	this book was just too much. i read the whole thing, but i didn't like the way the author ended it. i was hoping for a different ending.

Table 4: **U**: User. **P**: Product. **R**: Rating. This table shows some generated examples of the Att2Seq+A model. In every group, two attributes are kept unchanged, while the other attribute has different values. For instance, in the first group, we use different ratings ranging from 1 (the lowest score) to 5 (the highest score) with the same user and product to generate reviews. The users and products are anonymized by A-E and V-Z.

amples describe "*characters*" and "*story line*", and are written in the similar styles. This indicates that user and product information determines the content and style of generated reviews, while rating affects the choice of sentiment words. In the next group, we use different user IDs as input attributes. This book is one of the *Fatal Series* written by *Marie Force*, which tells a romantic love story. The first and third examples mention "*next one/book*", and both the first two reviews contain the phrase "*love story*". This demonstrates the generated reviews agree with the input product information. In the third group, the attributes, except product ID, are kept unchanged. The examples show our model generates varied reviews for different products.

5 Conclusion

In this paper, we proposed a novel product review generation task, in which generated reviews are conditioned on input attributes. For this task, we formulated a neural network based attribute-to-sequence model that uses multilayer perceptrons to encode input attributes and employs recurrent neural networks to generate reviews. Moreover, we introduced an attention mechanism to better utilize input attribute information. Additionally, we built a dataset of Amazon product reviews to conduct evaluations. The proposed model consistently outperforms the nearest neighbor search and maximum entropy language model baselines. Besides, the attention mechanism significantly improves the vanilla attribute-to-sequence model. This work suggests several interesting directions for future research. We could use more fine-grained attributes as the input of our model. For example, the generated reviews could be conditioned on device specification, brand, user's gender, product description, or ratings of a product's various aspects. Moreover, we could leverage review texts without attributes to improve the sequence decoder.

Acknowledgments

The support of the European Research Council under award number 681760 "Translating Multiple Modalities into Text" is gratefully acknowledged.

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*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, T. Darrell, and K. Saenko. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2625–2634.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany. Association for Computational Linguistics.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2014. Adaptive multi-compositionality for recursive neural models with applications to sentiment analysis. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 1537–1543. AAAI Press.

A. Dosovitskiy, J. T. Springenberg, and T. Brox. 2015. Learning to generate chairs with convolutional neural networks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1538–1546.

M. Gutmann and A. Hyvrinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Y.W. Teh and M. Titterington, editors, *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 9 of *JMLR WCP*, pages 297–304. Journal of Machine Learning Research - Proceedings Track.

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

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA. Association for Computational Linguistics.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2016. Visualizing and understanding recurrent networks. In *International Conference on Learning Representations*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Ioannis Konstas and Mirella Lapata. 2012. Concept-to-text generation via discriminative reranking. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 369–378, Jeju Island, Korea. Association for Computational Linguistics.

Ioannis Konstas and Mirella Lapata. 2013. A global model for concept-to-text generation. *Journal of Artificial Intelligence Research*, 48(1):305–346.

Zachary C. Lipton, Sharad Vikram, and Julian McAuley. 2015. Capturing meaning in product reviews with character-level generative text models. *arXiv preprint arXiv:1511.03683*.

Bing Liu. 2015. *Sentiment Analysis: Mining Opinions, Sentiments, and Emotions*. Cambridge University Press.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Umar Maqsud. 2015. Synthetic text generation for sentiment analysis. In *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 156–161, Lisboa, Portugal. Association for Computational Linguistics.

Julian McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 785–794, New York, NY, USA. ACM.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 720–730, San Diego, California. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Dae Hoon Park, Hyun Duk Kim, ChengXiang Zhai, and Lifan Guo. 2015. Retrieval of relevant opinion sentences for new products. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 393–402, New York, NY, USA. ACM.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of The 30th International Conference on Machine Learning*, pages 1310–1318.

Richard Socher, Cliff Chiung-Yu Lin, Andrew Ng, and Chris Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 129–136, New York, NY, USA. ACM.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

T. Tieleman and G. Hinton. 2012. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. Technical report.

Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2015. Sequence to sequence - video to text. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 4534–4542.

O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. 2015a. Show and tell: A neural image caption generator. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3156–3164.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015b. Grammar as a foreign language. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2773–2781. Curran Associates, Inc.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In David Blei and Francis Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2048–2057. JMLR Workshop and Conference Proceedings.

Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. 2016. Attribute2image: Conditional image generation from visual attributes. In *European Conference on Computer Vision*, pages 776–791.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2015. Recurrent neural network regularization. In *International Conference on Learning Representations*.