LexiCon: Lexically Constrained Review Generation via Robust Insertion

Anonymous ACL submission

Abstract

Existing review generators struggle to generate specific information correctly (e.g., *Caesar salad*, *Snapdragon CPU*), which prevents generated reviews from being more informative. In this paper, we propose to introduce lexical constraints into review generation which can be any key phrases to be contained in reviews. Compared to soft constraints (e.g., aspects) used in previous work, lexical constraints easily incorporate specific information which can largely improve the diversity and informativeness of generated reviews. To this end, we present LEXICON, a novel insertion-based review generation framework that can generate personalized reviews containing lexical constraints. Specifically, the proposed method progressively inserts new tokens between existing tokens in a parallel manner until a sequence is completed. Experimental results show that LEXICON outperforms the strongest review generation model by 20% BLEU-2 (coherence) and 68% Distinct-2 (diversity) on average. Human evaluation also shows that LEXICON is more robust to various lexical constraints than the state-of-the-art lexically-constrained model for general purpose.

1 Introduction

Personalized review generation models could work as (1) a writing tool (Li et al., 2021a) for *users* that assists the review writing process to encourage users providing their feedback; (2) an explanation generation system (Ni et al., 2019) from *businesses* that justifies the users' interests in a product by natural languages. The generated reviews have personalized writing styles and information on specific products by incorporating the product information and user behavior as input (Ni and McAuley, 2018; Zhou et al., 2017).

Previous works (Zhou et al., 2017; Wang and Zhang, 2017; Radford et al., 2017; Li and Tuzhilin, 2019) have explored the review generation task and shown success in generating cohesive reviews. Recent studies focus on increasing the controllability of the generation process so that the generated reviews will be more informative and relevant to users' interests. To this end, they use aspects extracted from data (Li et al., 2019; Ni and McAuley, 2018) or knowledge bases (Li et al., 2020a, 2021a) then apply text planning methods (Hua and Wang, 2019; Moryossef et al., 2019) to generate personalized reviews which describe products based on given information.

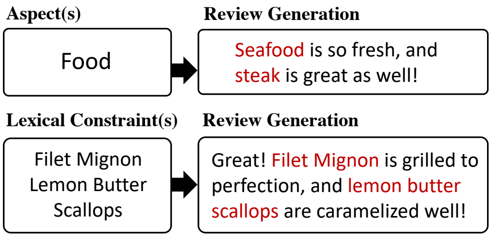


Figure 1: Example of reviews generated from Aspect-aware and Lexically-constrained methods.

However, existing review planning tools only have soft constraints (e.g. aspects) which mostly control the sentiment or semantics of generated text. In this case, users or businesses cannot conduct lexical manipulation of the generation process to have specific product attributes, but these attributes are too specific to be accurately generated. For example, as shown in Figure 1, a restaurant or user wants to include some featured dishes (`Filet Mignon` and `Lemon Butter Scallops`) into the explanations or reviews. Previous aspect-aware (soft-constraints) review generation methods control the generation process by giving an aspect `Food` but cannot ensure their dish names appear in the generated text. Moreover, generated dish names are usually general (`Seafood` and `Steak`). To show the missing key phrases in review generation, we have experiments (setup details in Appendix A) on comparing key phrase coverage (informativeness) between generated reviews and human-written re-

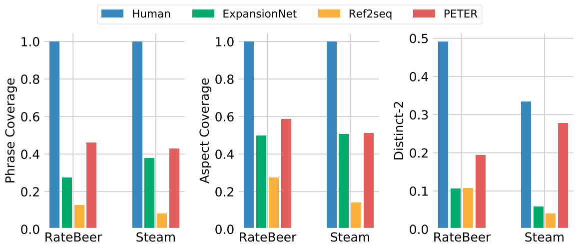


Figure 2: The phrase coverage, aspect coverage and Distinct-2 of generated reviews from ExpansionNet (Ni and McAuley, 2018), Ref2seq (Ni et al., 2019) and PETER (Li et al., 2021b) on RateBeer and Steam dataset.

views. Experimental results in Figure 2 show ExpansionNet largely improved the results (in Figure 2) compared to Ref2seq, but a lot of key phrases are still missed. Furthermore, with soft constraints only, existing methods also struggle to generate sufficiently diverse and informative reviews. The results in Figure 2 on key phrase coverage (informativeness), aspect coverage (controllability), and Distinct-2 (diversity) show that the strongest model (PETER) can cover at most 45% phrases and 60% aspects in reviews and the diversity (Distinct-2) of generated text is lower than reviews from human. Due to the limitation of current models, the generated reviews usually lack diversity and are rare to contain specific information about products.

To address the above problems, we propose a lexically constrained review generation task, in which the generated reviews must contain lexical constraints from users, businesses or even randomly sampled product attributes. Compared to previous methods with soft constraints that generate some general words (e.g., `Seafood`), lexically constrained review generation can easily incorporate specific information (e.g., `Lemon Butter Scallops`) into reviews. Hence, the informativeness and diversity of generated reviews can be largely improved (see Section 4).

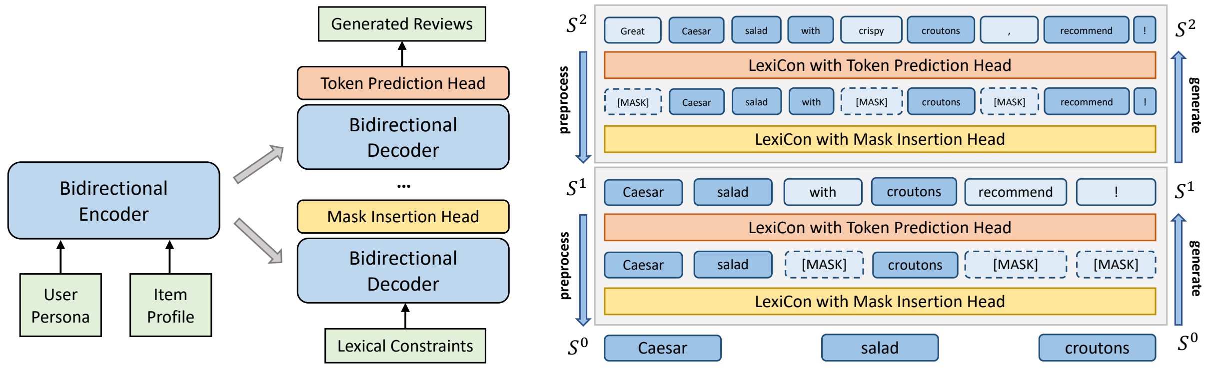
Existing lexically constrained text generation methods (Zhang et al., 2020b; Welleck et al., 2019; Miao et al., 2019) for general purpose cannot be directly applied to review generation due to three reasons: (1) special-decoding based methods (Hokamp and Liu, 2017; Post and Vilar, 2018; Hu et al., 2019; Miao et al., 2019) tend to have high complexity (Zhang et al., 2020b) at inference time and are not feasible for online services; (2) insertion based methods, such as POINTER (Zhang et al., 2020b), are not robust to arbitrary keywords that are not extracted by their pre-defined algorithm (see Section 4.6); (3) current methods focus on generating text from keywords but cannot incorporate personalized information, though reviews are usually personalized and contain different product features.

Motivated by the above, we propose a novel insertion-based framework for lexically constrained review generation, called LEXICON (**LEXI**cally **CON**strained review generation). Compared to existing lexically constrained methods, this framework is robust to arbitrary constraints and incorporates contextual information by an encoder so that LEXICON largely improves *relevance*, *coherence* and *informativeness* of generated reviews compared to existing methods. The main contributions of this paper are summarized as follows:

- To further improve the controllability and informativeness of review generation, we propose a lexically constrained review generation task, in which specific information can be easily contained in the generated reviews.
- We present LEXICON, an insertion-based framework which can generate personalized reviews from arbitrary lexical constraints. A large-scale pre-training is performed for downstream review generation tasks.
- We conduct extensive experiments on four review datasets. Objective metrics and human evaluations show that LEXICON can largely improve the diversity and informativeness of generated reviews, and our insertion process is more robust to lexical constraints than previous methods.

2 Related Work

Many attempts have been made to generate reviews for users. RNN-based methods (Tang et al., 2016) have been applied to generate the reviews with useful context information from users and items. Zhou et al. (2017) proposed an attribute-to-sequence (Attr2Seq) method to encode user and item identities with embeddings and then decode with LSTM to generate reviews. Some studies (Ni et al., 2017; Wang and Zhang, 2017; Li et al., 2020b) proposed to combine rating prediction and review generation, and utilize user-item interactions to improve the sentiment of generated reviews. To better control the review generation process, previous methods (Ni and McAuley, 2018; Li et al., 2019) extracted aspects and controlled the semantics of generated reviews conditioned on



(a) LexiCon model structure for review generation

(b) Data preprocess and text generation

Figure 3: The overview of (a) LexiCon model structure for review generation, and (b) training data construction and text generation process.

different aspect. Another line of work (Li et al., 2021a, 2020a) controlled and enriched generated reviews by knowledge bases. Although previous works continued increasing their controllability on generated reviews, they still struggle to have fine-grained manipulation such as self-defined review keywords from users. In this paper, we propose lexical constrained review generation which largely increase the controllability, informativeness and interpretability of generated reviews.

Hard-constrained generation requires that generated text contain the lexical constraints. Early works usually involve special decoding methods. Hokamp and Liu (2017) proposed a lexical-constrained grid beam search decoding algorithm to incorporate constraints. Post and Vilar (2018) presented an algorithm for lexically constrained decoding with a reduced complexity in the number of constraints. Hu et al. (2019) further improved decoding by a vectorized dynamic beam allocation. Miao et al. (2019) introduced a sampling-based conditional decoding method, where the constraints are first placed in a template, then decoded words under a Metropolis-Hastings sampling. Special decoding methods usually need a high running time complexity. Recently, Zhang et al. (2020b) implemented hard-constrained generation with a $\mathcal{O}(\log n)$ time complexity by language model pre-training and insertion-based generation (Stern et al., 2019; Gu et al., 2019b; Chan et al., 2019; Gu et al., 2019a) used in machine translation. However, we found that POINTER is not robust to arbitrary lexical constraints due to the data construction based on dynamic programming. Hence, we propose LexiCon which can generate personalized reviews with arbitrary lexical constraints.

3 LexiCon

We describe the lexically constrained review generation task as follows. Given a user u, item i, several lexical constraints (e.g, phrases or keywords) $\mathrm{C} = \{c_1, c_2, \ldots, c_m\}$, and historical reviews R^u, R^i of u and i respectively, our goal is to generate a review $\mathrm{R}^{ui} = (w_1, w_2, \ldots, w_n)$ that maximizes the probability $P(\mathrm{R}^{ui}|u, i, \mathrm{C})$. Different from previous review generation tasks, our generated review R^{ui} has to exactly include all given lexical constraints c_i, which means $c_i = (w_j, \ldots, w_k)$. The lexical constraints can be from users, businesses, or randomly selected from item attributes in a real application. In this paper, we extract noun phrases from reviews and use extracted phrases as lexical constraints.

3.1 Method Overview

The generation procedure of our method can be formulated as a progressive sequence of K stages $S = \{S^0, S^1, \ldots, S^{K-1}, S^K\}$, where S^0 is the stage of lexical constraints and S^K is our final generated text. For each $k \in \{1, \ldots, K\}$, S^{k-1} is a sub-sequence of S^k. The generation procedure finishes when LexiCon does not insert any new tokens into S^K.

Figure 3 (b) shows an example of our text generation process. For each insertion step from S^{k-1} to S^k, we decompose one step into two operations, mask insertion and token prediction. LexiCon first insert `[MASK]` tokens between any two existing tokens and the number of inserted `[MASK]` is predicted by an insertion head, then as other masked language models, our model predicts the word token for each `[MASK]`. Mask insertion and token prediction are both personalized by incorporating information from users and products with a text encoder. Overall, LEXICON has two components as shown in Figure 3 (a): (1) text encoder to incorporate user persona and item profile from historical reviews; (2) decoder with two different prediction heads, a token prediction head $\mathbf{H}_{TP}$ and mask insertion head $\mathbf{H}_{MI}$.

3.2 Data Preparation

For preparing training data, we construct pairs of text sequences at adjacent stages (S^{k-1}, S^k) that reverse the insertion-based generation process. Each review R^{ui} in the training data is broken into a consecutive series of pairs: $(S^0, S^1), (S^1, S^2), \ldots, (S^{K-1}, S^K)$, and when we construct the training data, the final stage S^K is our review text R^{ui}. In the previous work, POINTER (Zhang et al., 2020b) designed a method to compute the importance score of tokens and a dynamic programming algorithm to make sure that important tokens appear in an earlier stage and the number of stages K is small. However, we found that the model pre-trained by this method is sensitive to the initial lexical constraints S^0. If the constraint selections are not similar to the data preprocessing algorithm in POINTER training, the quality of generated reviews will decrease.

To alleviate the above problem, we propose a simple but effective data preparation method which makes the model robust to arbitrary lexical constraints. As illustrated in Figure 3 (b), given a sequence stage S^k, we obtain the previous stage S^{k-1} by two operations, masking and deleting. Specifically, we randomly mask the tokens in a sequence by probability p as masked language model pretraining (Devlin et al., 2019; Liu et al., 2019) to get the intermediate sequence $I^{k,k-1}$. Then, `[MASK]` tokens are deleted from the intermediate sequence $I^{k,k-1}$ to obtain the stage S^{k-1}. The numbers of deleted `[MASK]` tokens after each token in $I^{k,k-1}$ are recorded as an insertion number sequence $J^{k,k-1}$. Finally, each training instance contains four sequences $(S^{k-1}, I^{k,k-1}, J^{k,k-1}, S^k)$. Since we delete $T * \mathrm{p}$ tokens in sequence S^k where T is the length of S^k, the average number of K is $\log_{\frac{1}{1-\mathrm{p}}} T$.

3.3 Model Architecture

As shown in Figure 3 (a), LEXICON uses the sequence-to-sequence Transformer architecture with two different prediction heads for mask insertion and token prediction, but different from standard Transformer (Vaswani et al., 2017), our decoder is a bidirectional self-attention structure as encoder since LEXICON is a non-auto-regressive generation model. The architectures of the encoder and decoder are closely related to that used in RoBERTa (Liu et al., 2019), but each layer of the decoder additionally performs cross-attention over the final hidden layer of the encoder.

Context Encoder. Given a preprocessed training instance $(S^{k-1}, I^{k,k-1}, J^{k,k-1}, S^k)$ which is constructed from review R^{ui} with the method introduced in Section 3.2, we use user historical reviews R^u and item historical reviews R^i as our contextual information Φ^{ui}. R^u and R^i are concatenated by a special token `[SEP]` and the bidirectional encoder $\mathbf{E}$ encodes the concatenated reviews to get contextual information of user u and item i. Formally, the output is calculated as:

$$\mathbf{h}^{ui} = \mathbf{E}(\Phi^{ui}) = \mathbf{E}([\mathrm{R}^u; \mathrm{R}^i]) \tag{1}$$

where $[;]$ denotes the concatenation, $\mathbf{h}^{ui} \in \mathbb{R}^{t \times d}$, t is the length of concatenated reviews and d is the hidden size of our model.

Decoder with Two Heads. We decode the contextual information $\mathbf{h}^{ui}$ and existing token stage S^{k-1} with a bidirectional decoder $\mathbf{D}$. The decoder will predict the mask insertion numbers and word tokens with two heads $\mathbf{H}_{MI}$ and $\mathbf{H}_{TP}$ respectively. $\mathbf{H}_{TP}$ is a multilayer perceptron (MLP) with activation function GeLU (Hendrycks and Gimpel, 2016) and $\mathbf{H}_{MI}$ is a linear projection layer. Finally, our predictions of mask insertion numbers and word tokens are computed as:

$$y_{MI} = \mathbf{H}_{MI}(\mathbf{D}(S^{k-1}, \mathbf{h}^{ui})) \tag{2}$$

$$y_{TP} = \mathbf{H}_{TP}(\mathbf{D}(I^{k,k-1}, \mathbf{h}^{ui})) \tag{3}$$

where $\mathbf{h}^{ui}$ is incorporated by the cross attention of decoder, $y_{MI} \in \mathbb{R}^{l_s \times d_{ins}}$ and $y_{TP} \in \mathbb{R}^{l_I \times d_{vocab}}$. l_s and l_I are the length of S^{k-1} and $I^{k,k-1}$ respectively. d_{ins} is the maximum number of insertion and d_{vocab} is the size of vocabulary.

3.4 Model Training

The training process of LEXICON is to learn the inverse process of data generation. Given stage pairs (S^{k-1}, S^k) from user u, item i, and corresponding contextual information Φ^{ui} and training instance $(S^{k-1}, I^{k,k-1}, J^{k,k-1}, S^k)$ from preprocessing, we optimize the following objective:

$$
\begin{aligned}
\mathcal{L} &= -\log p(S^k|S^{k-1}, \Phi^{ui}) \\
&= -\log p(S^k|J^{k,k-1}, S^{k-1}, \Phi^{ui}) p(J^{k,k-1}|S^{k-1}, \Phi^{ui}) \\
&= -\log \underbrace{p(S^k|I^{k,k-1}, \Phi^{ui})}_{\text{Token prediction probability}} \underbrace{p(J^{k,k-1}|S^{k-1}, \Phi^{ui})}_{\text{Mask insertion probability}}, \\
I^{k,k-1} &= \mathrm{MaskInsert}(J^{k,k-1}, S^{k-1})
\end{aligned} \tag{4}
$$

where MaskInsert denotes the mask token insertion. In Equation (4), we jointly learn (1) likelihood of mask insertion number for each token from LEXICON with $\mathbf{H}_{MI}$, and (2) likelihood of word tokens for the masked tokens from LEXICON with $\mathbf{H}_{TP}$.

Same as training in BERT (Devlin et al., 2019), we optimize only the masked tokens in token prediction. The selected tokens to mask have the probability 0.1 to stay unchanged and probability 0.1 to be randomly replaced by another tokens in the vocabulary. For mask insertion number prediction, most numbers in $J^{k,k-1}$ are 0 because we do not insert any tokens between existing two tokens in most cases. To balance the insertion number, we randomly mask the 0 in $J^{k,k-1}$ by probability q.

Because our mask prediction task is similar to masked language models, the pre-trained weights from RoBERTa (Liu et al., 2019) can be naturally used for initialization of encoder and decoder in LEXICON to obtain prior knowledge. Moreover, we pre-train LEXICON on a massive review corpus for various domains to obtain pre-trained model that can be finetuned on downstream review generation tasks.

3.5 Inference

At inference time, we start from the given lexical constrain S^0 and use LEXICON predict $\{\hat{S}^1, \ldots, \hat{S}^K\}$ repeatedly until no additional tokens generated or reaching the maximum stage number. $\hat{S}^K$ is the final generated content.

Without loss of generality, we show the inference details from $\hat{S}^{k-1}$ stage[1] to $\hat{S}^k$ stage: (1) given $\hat{S}^{k-1}$ LEXICON uses $\mathbf{H}_{MI}$ to predict $\hat{J}^{k,k-1}$ insertion number sequence[2]; (2) given $\hat{I}^{k,k-1}$ from $\mathrm{MaskInsert}(\hat{J}^{k,k-1}, \hat{S}^{k-1})$, LEXICON can use $\mathbf{H}_{TP}$ to predict $\hat{S}^k$ with a specific decoding strategy such as greedy search or top-K sampling. (3) given $\hat{S}^k$, LEXICON meets the termination requirements or executes step (1) again.

[1]Or S^{k-1} stage when $k = 1$, we use $\hat{S}^{k-1}$ for simplicity.

[2]We set predicted insertion number as 0 for given phrases in S^0, to prevent given phrases from modification.

Dataset	Train	Dev	Test	#Users	#Items
RateBeer	16,839	1,473	912	4,385	6,183
GoodReads	385,369	10,394	8,655	18,147	182,501
Yelp	252,087	37,662	12,426	235,794	22,412
Steam	450,631	67,367	24,827	403,942	1,993

Table 1: Statistics of our datasets

4 Experiments

4.1 Datasets

For *pre-training*, we combined the reviews from Amazon[3] and Google Locals[4] with 55 million reviews. After data construction, the total number of training instances is up to 250 million; and for *fine-tuning*, we used four smaller reviews datasets in specific domain to evaluate our model, which are Yelp[5], RateBeer (McAuley and Leskovec, 2013), Steam (Pathak et al., 2017), and Goodreads (Wan and McAuley, 2018). We further filter the reviews with length is larger than 64. For each user, following Ni et al. (2019), we randomly hold out two samples from all of their reviews to construct the development and test sets.

4.2 Baselines

For automatic evaluation, we consider two groups of baselines to evaluate our model effectiveness, where we make the input constraints for baselines as the same as what we used in LEXICON, more details can be checked in Appendix B. The first group is existing personalized review generation models with *soft constraints*, which means models use lexical constraints as contextual information for review generation, but don't guarantee these specific lexical constraints appear in generation.

- **ExpansionNet** (Ni and McAuley, 2018), generates reviews conditioned on different aspects extracted from a given review title or summary.
- **Ref2Seq** (Ni et al., 2019), a Seq2Seq model incorporates contextual information from historical reviews and uses fine-grained aspects to control review generation..
- **PETER** (Li et al., 2021b), a Transformer-based model that uses user- and item-IDs and given phrases to predict the words in target personalized review generation.

The second group includes general controllable natural language generation models with *hard con-*

[3]https://www.amazon.com/

[4]https://www.google.com/maps

[5]https://www.yelp.com/dataset

	RateBeer							Yelp						
Model	B-2	B-4	M	R-L	BS	D-1	D-2	B-2	B-4	M	R-L	BS	D-1	D-2
Human-Oracle	–	–	–	–	–	8.3	49.2	–	–	–	–	–	3.8	34.1
ExpansionNet	8.5	1.6	29.8	24.4	81.9	0.9	10.6	7.1	1.1	28.5	21.0	82.7	0.7	9.3
Ref2Seq	13.8	2.9	28.8	32.7	86.6	1.5	10.7	4.4	0.3	14.7	18.3	84.6	0.5	8.1
PETER	25.4	9.4	40.0	40.8	87.0	2.4	19.4	20.7	5.5	38.6	44.2	87.2	2.0	21.6
NMSTG	5.7	0.8	31.9	30.0	85.5	8.9	69.0	4.8	0.3	26.2	27.6	84.1	**4.8**	64.0
POINTER	5.2	0.1	34.4	36.2	85.2	3.7	32.4	3.0	0.1	31.5	28.1	85.2	1.0	12.1
LexiCon	**35.1**	**15.8**	**54.6**	**60.8**	**91.0**	**8.6**	**44.0**	**22.1**	**7.7**	**43.8**	**51.4**	**89.5**	5.0	**36.1**

	Goodreads							Steam						
Model	B-2	B-4	M	R-L	BS	D-1	D-2	B-2	B-4	M	R-L	BS	D-1	D-2
Human-Oracle	–	–	–	–	–	6.9	39.1	–	–	–	–	–	4.3	33.5
ExpansionNet	3.1	0.6	21.8	13.7	78.8	0.6	3.9	3.7	0.8	21.9	11.4	75.7	0.9	5.9
Ref2Seq	10.6	2.2	24.8	29.6	85.5	2.3	13.4	3.5	0.3	11.9	15.6	82.4	0.2	4.1
PETER	18.6	5.7	35.4	42.8	85.2	4.1	24.7	19.8	6.1	38.4	46.0	85.7	2.9	27.0
NMSTG	4.0	0.2	24.0	23.3	82.5	9.5	70.9	7.2	1.0	26.7	30.6	84.7	**4.0**	55.1
POINTER	4.0	0.1	28.4	27.7	85.1	2.0	17.1	1.8	0.0	24.6	23.0	83.2	1.3	13.4
LexiCon	**21.6**	**7.8**	**41.4**	**48.7**	**89.1**	**8.9**	**39.3**	**22.4**	**8.3**	**42.0**	**47.1**	**88.7**	5.1	**32.7**

Table 2: Performance on automatic evaluation. The highest scores are **bold**. For Distinct metrics (D-1 and D-2), the scores closest to human-oracle are **bold**.

straints. We use two baselines,

- **NMSTG** (Welleck et al., 2019), a tree-based text generation scheme that from given lexical constraints in prefix tree form, the model generates words to its left and right, yielding a binary tree.
- **POINTER** (Zhang et al., 2020b), an insertion-based generation method pretrained on constructed data based on dynamic programming. We train our model based on a pre-trained model released from the authors.

Note that these baselines are proposed for general natural language generation without incorporating user personalized information. Other insertion-based transformers such as Insertion Transformer (Stern et al., 2019) and Levenshtein Transformer (Gu et al., 2019b) focus on machine translation instead of hard-constrained generation, so we do not consider them into our comparison.

For human evaluation, we choose the state-of-the-art review generation models PETER (Li et al., 2021b) and hard-constrained generation model POINTER (Zhang et al., 2020b) to make comparison with our LexiCon generated results.

4.3 Evaluation Metrics

Following Ni et al. (2019); Zhang et al. (2020b), we perform automatic evaluation with commonly-used text generation metrics including n-gram metrics including BLEU (B-1 and B-2) (Papineni et al., 2002), METEOR (M) (Banerjee and Lavie, 2005) and ROUGE-L (R-L) (Lin, 2004), diversity metric Distinct (D-1 and D-2) (Li et al., 2016). We also introduce BERT-score (BS) (Zhang et al., 2020a) as a semantic rather than n-gram metric.

4.4 Implementation Details

In training data construction, we randomly mask $\mathrm{p} = 0.2$ tokens in S^k to obtain $I^{k,k-1}$. The maximum length of concatenated reviews Φ^{ui} is set to 256. 0 in $J^{k,k-1}$ are masked by probability $\mathrm{q} = 0.9$. The structures of encoder and decoder are same as RoBERTa-base (Liu et al., 2019) and initialized with pre-trained weights. The tokenizer is byte-level BPE following RoBERTa. For *pre-training*, the learning rate is 5e-5, batch size is 512 and our model is optimized by AdamW (Kingma and Ba, 2015) in 1 epoch. For *fine-tuning* on down-stream tasks, the learning rate is 3e-5, batch size is 128 with the same optimizer as pre-training. The training epoch is 10 and we select the best model performing on the development set as our final model evaluated on test data. The lexical constraints are phrases extracted by spaCy [6] noun chunks.

[6] https://spacy.io/

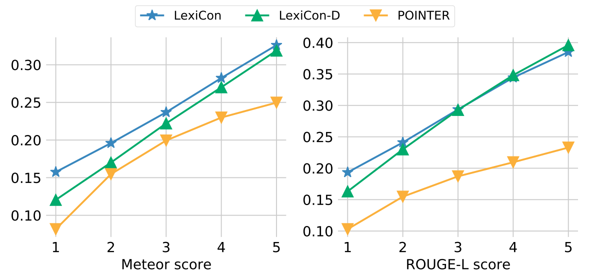
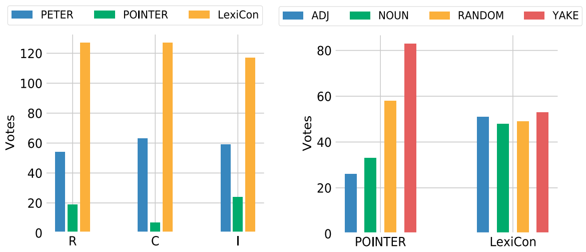


Figure 4: Meteor and Rouge-L scores on RateBeer dataset with different numbers (from 1 to 5) of lexical constraints.



(a) Review Generation Quality (b) Generation Robustness

Figure 5: Human evaluation results on (a) review generation quality; (b) generation robustness.

4.5 Automatic Evaluation

In Table 2, we report evaluation results of review generation on four review datasets in terms of n-gram metrics (BLEU, Meteor and ROUGE-L), semantic metric (Bert Score) and diversity metric (Distinct). Overall, LEXICON achieves the highest n-gram and semantic scores on four datasets consistently, which confirms that our model is able to generate the most relevant reviews. Specifically, we analyze the results from two aspects: (1) compared with review generation baselines, LEXICON improves the state-of-the-art model (PETER) by 20% BLEU-2 and 68% Distinct-2 on average. Although ExpansionNet is able to include more key phrases compared to Ref2seq (see analysis in Section 1), it cannot achieve more diverse reviews (Distinct) than Ref2seq due to the limitation of RNN-based sequence-to-sequence models. The results indicate that though lexical constraints are given, the existing review generation models with soft constraints still struggle to include specific information into generated reviews; (2) compared with lexically constrained generation baselines, LEXICON largely improves the coherence (53.5% Meteor and 81.6% ROUGE-L in average) of generated reviews than POINTER. The diversity of LEXICON generation is the closest to the human reviews. NMSTG has much higher diversity because it tends to insert less-related tokens with users or products. The results indicate the necessity of contextual encoder and previous methods are not robust to our lexical constraints. We have further validation in the next section.

Contextual Encoder. To show the effectiveness of our contextual encoder and the quality of generated reviews with different numbers of lexical constraints. We pre-train a model (denoted as LEXICON-D) using the same method introduced in Section 3 but do not incorporate contextual information with an encoder. Then, we evaluate the generated reviews of LEXICON, LEXICON-D and POINTER by giving different numbers of lexical constraints on RateBeer dataset. The results is shown in Figure 4. We can observe that (1) LEXICON outperforms LEXICON-D consistently on Meteor score, and on ROUGE-L when we provide fewer lexical constraints; (2) the quality of generated reviews increases while we give more constraints. (3) LEXICON and LEXICON-D outperform POINTER largely. POINTER cannot achieve the same improvement as the number of constraints increases. The results indicate that our encoder can provide contextual information to improve review generation. The contextual encoder is necessary especially when the number of constrains is small. As the number of constraints increases, LEXICON-D has similar scores as LEXICON because the lexical constraints can provide enough information to generate reviews.

4.6 Human Evaluation

To further validate our results in automatic metrics, we conduct human evaluation (details in Appendix C) on *review quality* and *generation robustness* to compare LEXICON and our baselines.

Review Quality. We evaluate the quality from three aspects of generated reviews: (1) *Relevance* (R) measures whether the generated output contains information relevant to the human review; (2) *Coherence* (C) measures whether generated reviews are logical, well-organized and easy to understand by humans; (3) *Informativeness* (I) measures how distinct the generated reviews are and how much specific information is included. Annotators select the best generated review from each aspect (details in Appendix C.1). The review quality evaluation results (see Figure 5 (a)) show that the generation

Model	Result from RateBeer	Model	Result from Yelp
Phrases	best sold beer, Barcelona, water	Phrases	Overpriced sushi, 55 bucks, crap, fridge, days
Human	This is best sold beer in Barcelona. Very cheap (more than water trust me!).	Human	Overpriced sushi, I paid 55 bucks for crap that taste like its been sitting in a fridge for days.
Ref2Seq	this is one of the best beers i have ever had. it's a good beer.	Ref2Seq	this place has been pretty good for the last times i've been to. i've been here a few times and i'm not sure why
PETER	this is the best beer i've ever tasted. it's got to be a beer to be the i have ever tasted. it's not bad but it's not bad for the price.	PETER	overpriced sushi i spent $300, i was told that they were closed for a crap fridge for days and they were closed for days and they were closed.
POINTER	this is possibly one one of the best beers i have ever sold. i think this as a great beer and a great beer. barcelona, is probably one of the better water to drink .	POINTER	terrible. if you should have waited over 2 hours here for overpriced sushi: no more. then, for what can i should pay for 55 bucks. oh, if they had all of the same crap, a screw up stuff in your fridge and no more. seriously, how do you waste your days here?!
LEXICON	This is the best sold beer in Barcelona. Light and the water is refreshing.	LEXICON	Really bad service. Overpriced sushi. Over 55 bucks. Got the crap in the fridge for 2 days.

Table 3: Generated reviews from RateBeer and Yelp datasets. Lexical constraints (phrases) are highlighted in reviews. The reviews of LEXICON is cased because we use byte-level BPE following RoBERTa.

quality of LEXICON is largely better than PETER and POINTER on all aspects.

Generation Robustness. To compare the robustness to lexical constraints between POINTER and LEXICON, we give different types but the same number (5 in our experiments) of keywords to models and generate reviews. The keywords types include: adjectives (ADJ); nouns (NOUN); random words (RANDOM); and keywords extracted by YAKE (Campos et al., 2018) used in POINTER (YAKE). Our annotators are asked to select the most coherent sentences among generated reviews from different constraints (details in Appendix C.2). The evaluation results are shown in Figure 5 (b). We can see that: (1) for LEXICON, reviews from different keyword types have similar votes which indicates annotators struggle to select the best review and the quality of generated reviews is consistent from different keywords. (2) for POINTER, the votes of YAKE keywords are much higher than others (especially for ADJ) which means the quality of POINTER generation is sensitive to lexical constraints. Based on the above analysis, we can conclude that LEXICON largely improves the robustness of generation compared to POINTER.

4.7 Case Studies

We compare generated reviews from Ref2Seq, PETER, POINTER and LEXICON in Table 3. From examples, we can see that (1) Generated reviews from Ref2Seq are general and hard to cover some specific words (e.g., Barcelona) due to the limitation of RNN-based sequence-to-sequence models. (2) PETER adopts Transformer-based model which can have direct attention on lexical constraints. Hence, PETER can copy some words from constraints to the generated reviews but the copy process easily lead to repeated sentences in reviews. (3) POINTER can include lexical constraints but these phrases are broken into words. The generated reviews are not coherent because POINTER is not robust to lexical constraints presumably. (4) LEXICON can easily include specific information precisely from lexical constraints. The generated reviews are coherent and relevant to human reviews.

5 Conclusion

In this paper, we propose to have lexical constraints in review generation which can largely improve the informativeness and diversity of generated reviews by including specific information. To this end, we present LEXICON, a lexically constrained review generation framework which can easily include lexical constraints by inserting new tokens to generate coherent reviews. We conduct comprehensive experiments on review generation. Results show that LEXICON significantly outperforms previous review generation models and lexically constrained models in terms of informative and coherence. In addition, user studies indicate LEXICON is robust to arbitrary lexical constraints and generates high-quality reviews consistently.

Ethical Consideration

One main concern associated with review generation is that the model can be misused for generating spam reviews. These considerations largely follow from other works on review generation (and personalized language modeling in general). We also note that these are fundamental concerns with Natural Language Generation (as are issues of bias, toxic content, etc.). On the other hand, developing these models can help understand the behaviors and patterns in a spam review thus contribute to its detection. We also emphasize that our model is intended to be used in human-in-the-loop settings rather than automated generation, to minimize possible risks.

References

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *IEEvaluation@ACL*.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Mário Jorge, C. Nunes, and Adam Jatowt. 2018. Yake! collection-independent automatic keyword extractor. In *ECIR*.

William Chan, Nikita Kitaev, Kelvin Guu, Mitchell Stern, and Jakob Uszkoreit. 2019. Kermit: Generative insertion-based modeling for sequences. *ArXiv*, abs/1906.01604.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805.

Jiatao Gu, Qi Liu, and Kyunghyun Cho. 2019a. Insertion-based decoding with automatically inferred generation order. *Transactions of the Association for Computational Linguistics*, 7:661–676.

Jiatao Gu, Changhan Wang, and Jake Zhao. 2019b. Levenshtein transformer. In *NeurIPS*.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv: Learning*.

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. In *ACL*.

J. Edward Hu, Huda Khayrallah, Ryan Culkin, Patrick Xia, Tongfei Chen, Matt Post, and Benjamin Van Durme. 2019. Improved lexically constrained decoding for translation and monolingual rewriting. In *NAACL*.

Xinyu Hua and Lu Wang. 2019. Sentence-level content planning and style specification for neural text generation. In *EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and William B. Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *NAACL*.

Junyi Li, Siqing Li, Wayne Xin Zhao, Gaole He, Zhicheng Wei, Nicholas Jing Yuan, and Ji-Rong Wen. 2020a. Knowledge-enhanced personalized review generation with capsule graph neural network. *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*.

Junyi Li, Wayne Xin Zhao, Zhicheng Wei, Nicholas Jing Yuan, and Ji-Rong Wen. 2021a. Knowledge-based review generation by coherence enhanced text planning. *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Junyi Li, Wayne Xin Zhao, Ji-Rong Wen, and Yang Song. 2019. Generating long and informative reviews with aspect-aware coarse-to-fine decoding. In *ACL*.

Lei Li, Yongfeng Zhang, and L. Chen. 2020b. Generate neural template explanations for recommendation. *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*.

Lei Li, Yongfeng Zhang, and Li Chen. 2021b. Personalized transformer for explainable recommendation. In *ACL/IJCNLP*.

P. Li and Alexander Tuzhilin. 2019. Towards controllable and personalized review generation. In *EMNLP*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *ACL 2004*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Julian McAuley and Jure Leskovec. 2013. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. *Proceedings of the 22nd international conference on World Wide Web*.

Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. 2019. Cgmh: Constrained sentence generation by metropolis-hastings sampling. In *AAAI*.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-step: Separating planning from realization in neural data-to-text generation. *ArXiv*, abs/1904.03396.

Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *EMNLP*.

Jianmo Ni, Zachary Chase Lipton, Sharad Vikram, and Julian McAuley. 2017. Estimating reactions and recommending products with generative models of reviews. In *IJCNLP*.

Jianmo Ni and Julian McAuley. 2018. Personalized review generation by expanding phrases and attending on aspect-aware representations. In *ACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Apurva Pathak, Kshitiz Gupta, and Julian McAuley. 2017. Generating and personalizing bundle recommendations on steam. *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Matt Post and David Vilar. 2018. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *NAACL*.

Alec Radford, Rafal Józefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *ArXiv*, abs/1704.01444.

Peter Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65.

Mitchell Stern, William Chan, Jamie Ryan Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. In *ICML*.

Jian Tang, Yifan Yang, Samuel Carton, Ming Zhang, and Qiaozhu Mei. 2016. Context-aware natural language generation with recurrent neural networks. *ArXiv*, abs/1611.09900.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *ArXiv*, abs/1706.03762.

Mengting Wan and Julian McAuley. 2018. Item recommendation on monotonic behavior chains. *Proceedings of the 12th ACM Conference on Recommender Systems*.

Zhongqin Wang and Yue Zhang. 2017. Opinion recommendation using a neural model. In *EMNLP*.

Sean Welleck, Kianté Brantley, Hal Daumé, and Kyunghyun Cho. 2019. Non-monotonic sequential text generation. In *ICML*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020a. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Yizhe Zhang, Guoyin Wang, Chunyuan Li, Zhe Gan, Chris Brockett, and Bill Dolan. 2020b. Pointer: Constrained progressive text generation via insertion-based generative pre-training. In *EMNLP*.

M. Zhou, Mirella Lapata, Furu Wei, Li Dong, Shaohan Huang, and Ke Xu. 2017. Learning to generate product reviews from attributes. In *EACL*.

A Motivating Experiment Details

In this experiment, we evaluate the diversity and informativeness of reviews. Specifically, we apply phrase coverage, aspect coverage and Distinct-2 to measure generated reviews and human-written-reviews.

For **phrase coverage**, we first extract noun phrases from reviews by spaCy [7] noun chunks. Then we compare the phrases in human-written reviews and generated reviews. If a phrase appears in both reviews, we consider it as a covered phrase by generated reviews. This experiment measures how many specific information can be included in the generated reviews.

For **aspect coverage**, we obtain the phrase embedding by first tokenize phrases into words and use averaged GloVe (Pennington et al., 2014) embeddings to represent a phrase. Then, we use K-means clustering algorithm to get the clusters of phrases and these clusters are viewed as aspects of reviews. To achieve the best clustering results, silhouette scores (Rousseeuw, 1987) are computed to find the best cluster numbers(in Table 4). Similar as prhases, if an aspect appears in both reviews, we consider it as a covered aspect by generated reviews. This experiment measures if the soft-constraints can control the semantics of generated reviews.

Datasets	RateBeer	Yelp	Steam	Goodreads
Aspect numbers	100	185	195	55

Table 4: Aspect numbers in datasets.

For **Distinct-2**, we use the numbers as described in Table 2.

B Baseline Details

For **ExpansionNet**, we use the default setting which uses hidden size 512 for RNN encoder and decoder, batch size as 25 and learning rate 2e-4. For soft constraints in ExpansionNet, we use the set of lexical constraints (as concatenated phrases) to replace the *title* or *summary* input as contextual information for training and testing.

For **Ref2Seq**, we use the default setting with 256 hidden size, 512 batch size and 2e-4 learning rate. For soft constraints, we concatenate our given phrases as reference (historical reviews are also incorporated as reference following the original implementation) as contextual information in training and testing.

For **PETER**, we use the original setting with 512 embedding size, 2048 hidden units, 2 self-attention heads with 2 transformer layers, 0.2 dropout. We use the training strategy suggested by the authors. Since original PETER only support single word as soft constraint, we adopt PETER to multiple words with maximum length of 20 and reproduced the original single-word model on our multi-word model. We input our lexical constraints as the multi-word input for PETER training and testing.

For **NMSTG**, we uses the default settings with an LSTM with 1024 hidden size with the uniform oracle. We convert our lexical constraints into a prefix sub-tree as the input of NMSTG, and then use the best sampling strategy in our testing (i.e., `StochasticSampler`) for NMSTG.

For **POINTER**, we use the pre-training `BERT-base` from WIKI to fine-tune 40 epochs on our downstreaming datasets. We use all the default setting except for batch sizes since POINTER requires 16 GPUs for distributed training that exceeds our computational resources. Instead, we train POINTER with the same configuration on 3 GPUs. For testing, we select the base maximum turn as 3 with default greedy decoding strategy. We feed lexical constraints as the original implementation.

C Human Evaluation Details

We conduct two human evaluation experiments on RateBeer and Yelp datasets: (1) **Quality Experiment**: to evaluate the generation quality of generated reviews; (2) **Robustness Experiment**: to evaluate the generation robustness with respect to lexical constraints from various sources, since we find POINTER is sensitive to initial lexical constraints as mentioned in Section 3.2.

C.1 Quality Experiment Setup

Question Design. We uniformly sample 200 ground truth reviews (GT) from RateBeer and Yelp datasets in total, then collect corresponding generated reviews from PETER, POINTER and LexiCon respectively. Given the GT, annotator is requested to select the *best* review on different aspects i.e., *relevance*, *coherence* and *informativeness* (explained in Section 4.6) among reviews generated from PETER, POINTER and LexiCon. Ta-

[7] `https://spacy.io/`

ble 5 is an example of our evaluation template.

GT: delicious breakfast plates ... pecan waffle which came in the shape of texas! pretty cool :)	R	C	I
food delicious fresh breakfast ! plates great and service excellent ... i the shape the toast texas !			
great place. delicious breakfast plates and great service. the pecan waffle is in great shape in texas.	✓	✓	
delicious breakfast plates. service was great. pecan waffle ... perfect. texas was very nice.			✓

Table 5: A simple example of quality experiment multiple-choice question. Annotator checks the best in terms of *relevance* (R), *coherence* (C) and *informativeness* (I).

Experiment Conduction. We uniformly split the samples for 5 annotators so there are 40 multiple-choice questions per annotator, noting that displaying order of generated reviews are shuffled every time. Those 5 annotators are volunteer students for this project without payment, they are consent the fully use of collected data in the experiments for this paper after reading our instructions.

C.2 Robustness Experiment Setup

Question Design. We uniformly sample 200 ground truth reviews from RateBeer and Yelp datasets in total, then extract corresponding initial lexical constraints from ADJ, NOUN, YAKE, RANDOM strategies (described in Section 4.6). For POINTER or LEXICON, we generate reviews from lexical constraints coming from those five strategies respectively. Then we group the four generated reviews from the same model and the same GT as a set. Annotator is asked to choose the most coherent generated review from this set without knowing which initial lexical constraints strategy it comes from. Table 6 is an example of our evaluation template.

Generation	Best
amazing service! i wish they had more options and a little different flavors, but overall well the best mongolian food!	
you get what you pay for the different combinations. prices are great. the ingredients is fresh and the food is a good value.	✓
great selection of veggies all the time. fresh and great combinations. fresh and a cold beer, soup and tea.	
food is amazing and the selection of the fish bowl delicious. lots of different combinations. love this place!	

Table 6: A simple example of robustness experiment multiple-choice question. Annotator checks the best without knowing the initial lexical constraints (ADJ, NOUN, YAKE or RANDOM).

Experiment Conduction. We uniformly split the samples for 5 annotators. Thus, there are 40 multiple-choice questions from POINTER and 40 multiple-choice questions from LEXICON per annotator. Those 5 annotators are volunteer students for this project without payment. They are consent the fully use of collected data in the experiments for this paper after reading our instructions.

D GPU Hours

Our pre-training model is trained on 3 NVIDIA Quadro RTX 8000 graphical cards with 48GiB memory for 13 days. Our fine-tuning models are trained on single NVIDIA Quadro RTX 8000 graphical card with 48 GiB memory for averagely 10 hours per dataset. We acknowledge that one limitation of our model is LEXICON is a pre-training model on large-scale datasets so it is heavy to train. But for downstream review generation domains, fine-tuning is much faster.

E Packages

SpaCy. We use `en_core_web_sm` pre-trained natural language pipeline to process our data. All other settings are default in this pre-trained pipeline.

NLTK. We use NLTK to compute BLEU scores and all settings are default.

Huggingface Datasets [8]. We use this package to compute Meteor, ROUGE-L and BERT score (RoBERTa model).

[8] https://huggingface.co/docs/datasets/