AutoTemplate: A Simple Recipe for Lexically Constrained Text Generation

Anonymous ACL submission

Abstract

Lexically constrained text generation is one of the constrained text generation tasks, which aims to generate text that covers all the given constraint lexicons. While the existing approaches tackle this problem using a lexically constrained beam search algorithm or dedicated model using non-autoregressive decoding, there is a trade-off between the generated text quality and the hard constraint satisfaction. We introduce AutoTemplate, a simple yet effective lexically constrained text generation framework divided into template generation and lexicalization tasks. The template generation is to generate the text with the placeholders, and lexicalization replaces them into the constraint lexicons to perform lexically constrained text generation. We conducted the experiments on two tasks: keywords-to-sentence generations and entity-guided summarization. Experimental results show that the AutoTemplate outperforms the competitive baselines on both tasks while satisfying the hard lexical constraints.

1 Introduction

Text generation often requires lexical constraints, i.e., generating a text containing pre-specified lexicons. For example, the summarization task may require the generation of summaries that include specific people and places (Fan et al., 2018; He et al., 2022), and advertising text requires the inclusion of pre-specified keywords (Miao et al., 2019; Zhang et al., 2020b).

However, the black-box nature of recent text generation models with pre-trained language models (Devlin et al., 2019; Brown et al., 2020) makes it challenging to impose such constraints to manipulate the output text explicitly. Hokamp and Liu (2017) and others tweaked the beam search algorithm to meet lexical constraints by increasing the weights for the constraint lexicons, but it often misses to include all the constrained lexicons. Miao et al. (2019) and others introduced specialized non-autoregressive models (Gu et al., 2018) that insert words between the constraint lexicons, but the generated texts tend to be lower-quality than standard autoregressive models.

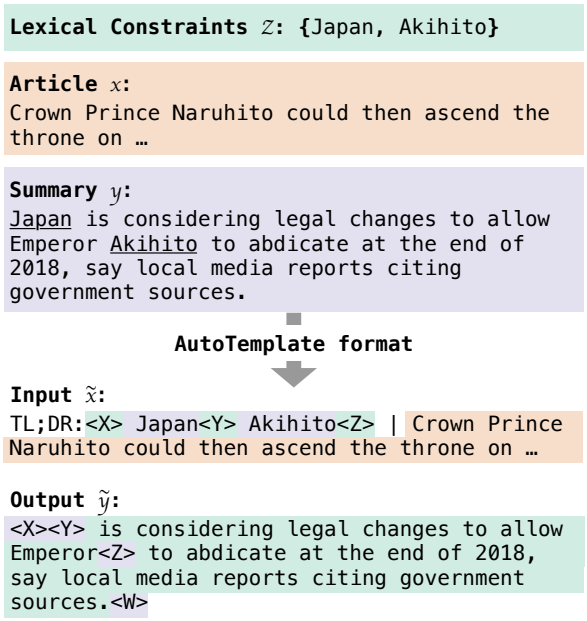

Figure 1: Illustration of AutoTemplate. We build the model input $\tilde{x}$ by concatenating the constraint lexicons $\mathcal{Z}$ with mask tokens. For the conditional text generation task, we further concatenate input document x. We also build the model output $\tilde{y}$ by masking the constraint lexicons in summary y. Then, we can train a standard sequence-to-sequence model, $p(\tilde{y} \mid \tilde{x})$, generate masked template $\tilde{y}$ given input $\tilde{x}$, and post-process to achieve lexically constrained text generation.

On the other hand, classical template-based methods (Kukich, 1983) can easily produce text that satisfies the lexical constraints as long as we can provide appropriate templates. Nevertheless, it is impractical to prepare such templates for every combination of constraint lexicons unless for specific text generation tasks where the output text patterns are limited, such as data-to-text generation tasks (Angeli et al., 2010). Still, if such a template could be *generated automatically*, it would be eas-

ier to perform lexically constrained text generation.

We propose AutoTemplate, a simple framework for lexically constrained text generations by automatically generating templates given constrained lexicons and replacing placeholders in the templates with constrained lexicons. The AutoTemplate, for example, can be used for summarization tasks, as illustrated in Figure 1, by replacing the constraint lexicons (i.e., {Japan, Akihito}) in the output text with placeholder tokens during training and using these constraints as a prefix of the input, creating input-output pairs, and then using a standard auto-regressive encoder-decoder model (Sutskever et al., 2014) to train the AutoTemplate model. During the inference, the constraint lexicons are prefixed in the same way, the model generates the template for the constraints, and the placeholder tokens are replaced with the constraint lexicons to perform lexically constrained text generation.

We evaluate AutoTemplate across two tasks: keywords-to-sentence generation on One-Billion-Words and Yelp datasets (§3.1), and entity-guided summarization on CNNDM (Hermann et al., 2015) and XSum datasets (Narayan et al., 2018) (§3.2). The AutoTemplate shows better keywords-to-sentence generation and entity-guided summarization performance than competitive baselines, including autoregressive and non-autoregressive models, while satisfying hard lexical constraints. We will release our implementation of AutoTemplate under a BSD license upon acceptance.

2 AutoTemplate

AutoTemplate is a simple framework for lexically constrained text generation (§2.1), divided into two steps: template generation (§2.2) and lexicalization (§2.3). The template generation task aims to generate the text with placeholders $\tilde{y}$, which we defined as a template, given constraint lexicons $\mathcal{Z}$, and the lexicalization is to replace these placeholders with the constraints to perform lexically constrained text generation.

2.1 Problem Definition

Let x be a raw input text, and $\mathcal{Z}$ be a set of constraint lexicons; the goal of the lexically constrained text generation is to generate a text y that includes all the constraint lexicons $\mathcal{Z}$ based on the input text x. For example, given a news article x and some entities of interest $\mathcal{Z}$, the task is to generate a summary y that includes all entities. Note that unconditional text generation tasks, such as keywords-to-sentence generation (§3.1), are only conditioned by a set of lexicons $\mathcal{Z}$, and in this case, we treat the input data x as empty to provide a unified description without loss of generality.

2.2 Template Generation

Given training input-output pairs (x, y) and constraint lexicons $\mathcal{Z}$, we aim to build a model that generates a template $\tilde{y}$, which has the same number of placeholder tokens as the constraint lexicons $\mathcal{Z}$. We assume that the output text y in the training set includes all the constraint lexicons $\mathcal{Z}$.

The template $\tilde{y}$ is created by replacing the constraint lexicon $\mathcal{Z}$ in the output text y with unique placeholder tokens according to the order of appearances (i.e., <X>, <Y>, and <Z> in Figure 1),[1] and then the model input $\tilde{x}$ is created by prefixing the constraint lexicons $\mathcal{Z}$ with the raw input text x.[2] These lexicons $\mathcal{Z}$ are concatenated with the unique placeholder tokens to let the model know the alignment between input and output. We discuss this design choice in §4.

Using the AutoTemplate input-output pairs $(\tilde{x}, \tilde{y})$, we can build an automatic template generation model $p(\tilde{y}|\tilde{x})$ using any sequence-to-sequence models. This study builds the template generation model p using an autoregressive Transformer model with a regular beam search (Vaswani et al., 2017).

2.3 Lexicalization

After generating the template $\tilde{y}$, we replace the placeholder tokens with constraint lexicons $\mathcal{Z}$ as post-processing to achieve lexically constrained text generation. Specifically, during inference, constraint lexicons are prefixed to the input text x in the same way to build the model input $\tilde{x}$. Then, we can obtain the template $\tilde{y}$ from the model p and replace the placeholder tokens with the constraint lexicons $\mathcal{Z}$.

2.4 Comparison with existing approaches

An important contribution of this study is to show that lexically-constrained generation can be performed in a simple way with AutoTemplate,

[1] We also prefix and postfix the placeholder tokens to use them as BOS and EOS tokens.

[2] We use | as separator token for constraints $\mathcal{Z}$ and input text x and also prefixed `TL;DR:`.

	multiple keywords	autoregressive decoding	keyword conditioning	constraint satisfaction
SeqBF (Mou et al., 2016)	✗	✗	✓	✓
CGMH (Miao et al., 2019)	✓	✗	✓	✓
GBS (Hokamp and Liu, 2017)	✓	✓	✗	✗
CTRLsum (He et al., 2022)	✓	✓	✓	✗
InstructGPT (Ouyang et al., 2022)	✓	✓	✓	✗
AutoTemplate (ours)	✓	✓	✓	✓

Table 1: Summary of existing work for lexically constrained text generation. SeqBF (Mou et al., 2016) and CGMH (Miao et al., 2019) use non-autoregressive decoding methods to insert words between given keywords. While these methods easily satisfy the lexical constraints, in general, non-autoregressive methods tend to produce lower-quality text generation than autoregressive methods. GBS (Hokamp and Liu, 2017), CTRLSum (He et al., 2022), and InstructGPT (Ouyang et al., 2022) use autoregressive methods to perform text generation, but there is no guarantee to satisfy all lexical constraints. AutoTemplate empirically demonstrates the capability to generate text that satisfies the constraints.

whereas it was previously done with only complicated methods. As summarized in Table 1, SeqBF (Mou et al., 2016) is the first neural text generation model for lexically constrained text generation based on non-autoregressive decoding. The SeqBF performs lexically constrained text generation by generating forward and backward text for a given constraint lexicon. The most significant limitation is that only a single keyword can be used for the constraint.

CGMH (Miao et al., 2019) and similar models (Zhang et al., 2020b; He, 2021) are yet another non-autoregressive models that achieve lexicon-constrained generation by inserting words between given constraint vocabularies, thus easily incorporating multiple constraints into the output text. Nevertheless, non-autoregressive models require complicated modeling and training to generate text as good as that of autoregressive models. We confirmed that the AutoTemplate produces consistently higher quality text than non-autoregressive methods, with or without leveraging pre-training (§3.1).

Another direction is to incorporate *soft* constraints into the autoregressive models such as constrained beam search (Hokamp and Liu, 2017; Post and Vilar, 2018) and keywords conditioning (He et al., 2022). GBS (Hokamp and Liu, 2017) is a constrained bean search technique that incorporates multiple keywords as constraints and promotes the inclusion of those keywords in the output during beam search. However, GBS often misses keywords in the output text.

CTRLSum (He et al., 2022) imposes keyword conditioning into encoder-decoder models by prefixing the keywords with the input. This method can be easily conditioned with multiple keywords as a prefix and can be implemented on an autoregressive model, resulting in high-quality text generation. However, the CTRLSum model cannot guarantee to satisfy lexical constraints. Our experiments show that as the number of constraints increases, it is more likely to miss constraint lexicons in the output text (§3.2).

InstructGPT (Ouyang et al., 2022) has shown remarkable zero-shot ability in many NLP tasks, and lexically constrained text generation is no exception. Our experiments confirmed that the model can generate a very fluent sentence, but as with CTRLSum, we observed a significant drop in the success rate with each increase in the number of keywords.

3 Experiments

We present experiments across two tasks: keywords-to-sentence generation (§3.1), and entity-centric summarization (§3.2).

3.1 Keywords-to-Sentence Generation

Keywords-to-sentence generation is a task to generate a sentence that includes pre-specified keywords as lexical constraints. We will show that AutoTemplate is a simple yet effective method to perform this problem without relying on any complex decoding algorithms.

Dataset We use One-Billion-Word and the Yelp dataset following the previous studies (Miao et al., 2019; Zhang et al., 2020b; He, 2021). One-Billion-Word is a dataset for language modeling based on the WMT 2011 news crawl data (Chelba et al., 2014). The Yelp dataset is based on the Yelp open dataset.[3] We utilized the publicly available pre-processed dataset,[4] which consists of 1M, 0.1M

[3] https://www.yelp.com/dataset
[4] https://github.com/NLPCode/CBART

Model	One-Billion-Word						Yelp					
	B2	B4	N2	N4	M	SR	B2	B4	N2	N4	M	SR
SeqBF (Mou et al., 2016)	4.4	0.7	0.62	0.62	7.0	<100.	6.9	2.1	0.52	0.53	8.7	<100.
GBS (Hokamp and Liu, 2017)	10.1	2.8	1.49	1.50	13.5	≤100.	13.6	4.5	1.68	1.71	15.3	≤100.
CGMH (Miao et al., 2019)	9.9	3.5	1.15	1.17	13.1	100.	12.3	4.6	1.41	1.45	14.6	100.
POINTER (Zhang et al., 2020b)	8.7	1.6	2.11	2.12	14.3	100.	10.6	2.4	2.14	2.16	16.8	100.
CBART (He, 2021)	15.6	6.6	2.16	2.19	15.2	100.	19.4	9.0	2.54	2.64	17.4	100.
InstructGPT (Ouyang et al., 2022)	10.1	2.8	1.72	1.73	13.0	92.33	9.3	2.4	1.42	1.44	13.6	92.17
AutoTemplate												
w/ T5-small	16.4	6.1	3.11	3.15	15.5	100.	22.5	9.5	3.51	3.63	17.1	100.
w/ T5-base	18.3	7.6	3.39	3.45	16.0	100.	23.7	10.8	3.62	3.76	17.8	100.
w/ T5-large	**18.9**	**8.1**	**3.49**	**3.54**	**16.2**	100.	**24.1**	**11.1**	**3.68**	**3.83**	**17.9**	100.

Table 2: Results of keywords-to-sentence generation on the One-Billion-Word and Yelp datasets. **Bold-faced** and underlined denote the best and second-best scores respectively. Baseline results are copied from He (2021). B2/4 denotes BLEU-2/4, N2/4 denotes NIST-2/4, M denotes METEOR-v1.5, and SR denotes the success rate of lexical constraint satisfaction. Non-aggregated results are shown in Table 17.

Data	# example	output len.	# constraints
1B-Words	12M	27.08	1 – 6
Yelp	13M	34.26	1 – 6
CNNDM	312k	70.58	4.53
XSum	226k	29.39	2.11

Table 3: Dataset Statistics: The output length is the number of BPE tokens per example using the T5 tokenizer. For the summarization datasets, the average number of constraints per example is shown.

sentences for training and development sets, respectively, and 6k sentences with 1-6 pre-specified keywords for test sets, which we summarized in Table 3.

Baselines For the baselines, we used strong competitive models for lexically constrained text generation, including SeqBF (Mou et al., 2016), GBS (Hokamp and Liu, 2017), CGMH (Miao et al., 2019), POINTER (Zhang et al., 2020b), CBART (He, 2021), and InstructGPT (Ouyang et al., 2022). SeqBF, GBS, and CGMH are implemented on top of GPT2-small (Radford et al., 2019) (117M parameters). POINTER is implemented on BERT-large (Devlin et al., 2019) (340M parameters), CBART is on BART-large (Lewis et al., 2020) (406M parameters), and InstructGPT has 175B parameters.[5]

Model We instantiate the template generation model based on the Transformer (Vaswani et al., 2017) initialized with T5 checkpoints (Raffel et al., 2020) implemented on `transformers` library (Wolf et al., 2020). We specifically utilized the T5-v1.1-small (60M), T5-v1.1-base (220M parameters), and T5-v1.1-Large (770M parameters). To train the model, we used AdamW optimizer (Loshchilov and Hutter, 2019a) with a linear scheduler and warmup, whose initial learning rate is set to 1e-5, and label smoothing (Szegedy et al., 2016) with a label smoothing factor of 0.1.

Since the dataset used in this experiment is a set of raw texts, we randomly select 1 to 6 words from the text and decompose them into constraint lexicons $\mathcal{Z}$ and a template $\tilde{y}$ to create the AutoTemplate training data. Note that the constraint lexicons $\mathcal{Z}$ were selected from the words excluding punctuations and stopwords (Loper and Bird, 2002).

Metrics All performance is measured with the BLEU-2/4 (Papineni et al., 2002), NIST-2/4 scores (Doddington, 2002), and METEOR v1.5 (Denkowski and Lavie, 2014). Following the previous study, we show the averaged performance across the number of keywords (He, 2021), but we also report the non-averaged results in Appendix.

Results Table 2 shows the results of keywords-to-sentence generation. First, the performance of GBS and InstructGPT is not as high as non-autoregressive methods. In general, autoregressive decoding produces better text quality than non-autoregressive decoding. However, since GBS is not conditioned on the keywords, it sometimes produces more general text that does not satisfy the keyword constraint. Also, InstructGPT tries to generate sentence according to the instructions, but our experiments show that it frequently fails to include constrained keywords.

Second, among the non-autoregressive baseline models, CBART outperforms CGMH and POINTER. This suggests that encoder-decoder-based models such as CBART can produce higher-quality text than decoder-only models such as CGMH and POINTER.

[5]Experimental details of InsturctGPT is in Appendix.

Keywords: leading , currency , software , industry
Reference: Transoft International , Inc. is a leading provider of currency supply chain management software solutions for the banking industry .
CBART: The leading edge currency trading software industry .
AutoTemplate: The company is a leading provider of currency management software to the financial services industry .

Table 4: Example generations for the keywords-to-sentence generation on One-billion-word.

Keywords: nail , salon , always , world
Reference: this is the very best nail salon ! i always see amanda , her workmanship is out of this world !
CBART: this is my favorite nail salon in town ! always clean , friendly and the world amazing .
AutoTemplate: I have been going to this nail salon for over a year now. they always do a great job, and the prices are out of this world .

Table 5: Example generations for the keywords-to-sentence generation on Yelp.

Finally, AutoTemplate consistently outperforms all the baselines on both datasets by a large margin while keeping the success rate at 100% regardless of the model size. This indicates that AutoTemplate could take advantage of both autoregressive decoding and encoder-decoder models as described above. We also confirm that using larger T5 models consistently improves text generation quality across all metrics.

Table 4 and 5 show qualitative examples of generated texts of CBART and AutoTemplate and human written reference. The examples show that the AutoTemplate generates long and fluent sentences while the CBART tends to generate short text in Table 4 or non-fluent text in Table 5. More examples can be found in Appendix.

3.2 Entity-guided Summarization

Automatic text summarization distills essential information in a document into short paragraphs, but different readers might want to know different things about specific entities, such as people or places. Thus, one summary might not meet all readers' needs. Entity-guided summarization aims to generate a summary focused on the entities of interest. This experiment demonstrates that AutoTemplate can produce summaries that satisfy lexical constraints, even under complex entity conditioning.

Dataset We use CNNDM dataset (Hermann et al., 2015) and XSum dataset (Narayan et al., 2018) for the experiment. We simulate the entity-guided summarization setting by providing the oracle entity sequence from the gold summary as lexical constraints. Specifically, we use stanza, an off-the-shelf NER parser (Qi et al., 2020), to parse the oracle entity sequence from the gold summary to create entity-guided summarization data. As summarized in the statistics in Table 3 and more detailed entity distributions in Figure 2, the CNNDM dataset tends to have more entities than the XSum dataset. Note that one instance in the test set of the CNNDM dataset has a 676-word reference summary with 84 oracle entities, which is difficult to deal with large pre-trained language models, so we excluded it from the success rate evaluation.

Baselines We used competitive models as baselines, including fine-tuned BART (Lewis et al., 2020) and CTRLSum (He et al., 2022). Similar to AutoTemplate, CTRLSum further conditions the input with lexical constraints and generates the output. The difference is that CTRLSum directly generates the output text, while AutoTemplate generates the corresponding template.

Model We use the same training configurations to instantiate the model used in the keywords-to-sentence generation task. To build the training dataset, we use the masked gold summary by the oracle entity sequence as the output template $\tilde{y}$ as described in §2, At inference time, we use the oracle entity sequence and the source document as input to generate the template and post-process to produce the output summary.

Metrics We evaluate the entity-guided summarization performance using F1 scores of ROUGE-1/2/L (Lin, 2004),[6] BERTScore (Zhang et al., 2020a),[7] and the success rate of entity constraint satisfaction. Note that our evaluation protocol for the success rate of entity constraint satisfaction is

[6] https://github.com/pltrdy/files2rouge
[7] https://github.com/Tiiiger/bert_score

Model	CNNDM					XSum				
	R1	R2	RL	BS	SR	R1	R2	RL	BS	SR
reported results										
BART (Lewis et al., 2020)	44.24	21.25	41.06	0.336	-	45.14	22.27	37.25	-	-
CTRLSum (He et al., 2022)	48.75	25.98	45.42	0.422	-	-	-	-	-	-
our implementation										
BART (Lewis et al., 2020)	44.20	21.28	41.02	0.358	26.12	44.21	20.93	35.18	0.510	46.69
CTRLSum (He et al., 2022)	47.57	25.56	44.30	0.437	75.46	50.07	26.73	40.90	0.581	86.32
AutoTemplate	.									
w/ T5-base	51.02	27.59	47.85	0.441	100.	50.49	28.19	43.89	0.591	100.
w/ T5-large	**52.56**	**29.33**	**49.38**	**0.465**	100.	**52.65**	**30.52**	**46.19**	**0.614**	100.

Table 6: Results of entity-guided summarization with oracle entities on CNNDM and XSum datasets. R1/2/L denotes ROUGE-1/2/L, BS denotes BERTScore, and SR denotes the success rate of lexical constraint satisfaction. **Bold-faced** and underlined denote the best and second-best scores respectively.

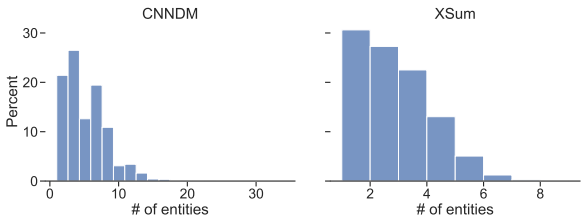

Figure 2: Distribution of the number of oracle entities. The CNNDM dataset (left) tends to have longer summaries and contains more entities than the XSUM dataset. As the number of entities increases, it becomes more and more difficult to include all the entities in the generated summary.

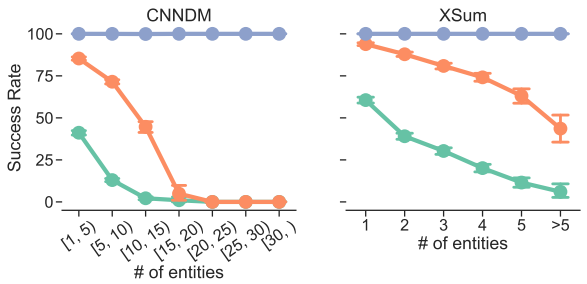

Figure 3: Success rate of entities included in the generated summary at a different number of entities. The **green line** denotes the BART model (Lewis et al., 2020), the **orange line** denotes the CTRLSum model (He et al., 2022), and **blue line** denotes AutoTemplate model. These graphs show that CTRLSum can include a limited number of entities in summary with a high chance. However, it becomes more and more difficult as the number of entities increases, while AutoTemplate always satisfies the constraint.

different and more difficult than in previous studies. (Fan et al., 2018; He et al., 2022). While the previous studies measure whether a *single* specified entity is included in the generated summary, this study measures whether *all* oracle entities are included.

Results Table 6 shows the results of entity-guided summarization. CTRLSum and AutoTemplate show improvements in summarization performance compared to the standard BART model, indicating that entity guidance contributes to the improvement in summarization performance.

On the other hand, while AutoTemplate always satisfies entity constraints, CTRLSum shows a constraint satisfaction success rate of 75.46% for CNNDM and 86.32% for XSum, characterizing the difference between AutoTemplate and CTRLSum. As shown in Figure 3, while CTRLSum shows a high success rate when the number of entity constraints is limited, the success rate decreases monotonically as the number of constraints increases. In contrast, the AutoTemplate showed a 100% success rate regardless of the number of entity constraints and the highest summarization quality.

Table 7 shows the qualitative examples of the generated summaries by CTRLSum and AutoTemplate. While CTRLSum could only include 10 of the 18 constraint entities in the generated summary, AutoTemplate covered all entities and generated a fluent summary.

We also show the generated summaries with different entity conditioning by AutoTemplate in Table 8. We confirmed that AutoTemplate can produce summaries with a different focus using different entity conditioning and can also include constraint entities in the generated summary.

4 Analysis

Does AutoTemplate generate fluent text? AutoTemplate decomposes the lexically constrained text generation task into template generation and

Constrained Entities: { Amir Khan , Manny Pacquiao , Abu Dhabi , UAE , Khan , Floyd Mayweather Jr , Las Vegas , PacMan , Bob Arum , UAE , Khan , Muslim , Brit , the Money Man , PacMan , Khan , Chris Algieri , New York }
CTRLSum (He et al., 2022): Amir Khan could face Manny Pacquiao in Abu Dhabi , UAE . Khan has been linked with a fight with Floyd Mayweather Jr in Las Vegas . The PacMan 's promoter Bob Arum is keen for a fight in the UAE .
AutoTemplate: Amir Khan could face Manny Pacquiao in Abu Dhabi , UAE . Khan is preparing to face Floyd Mayweather Jr in Las Vegas on May 2. PacMan 's vintage promoter Bob Arum has to hand a treasure trove of an offer for a fight in the UAE this November or December. Khan is a hero of the Muslim world, the Brit would be a huge attraction there. Assuming that the Money Man wins his interim bout with PacMan next month, all that would appear to stand between him and his long-awaited mega-fight is the outside chance of a re-match. Khan is set to fight Chris Algieri in New York next month.

Table 7: Qualitative comparisons between CTRLSum and AutoTemplate. Constraint entities are extracted from the reference summary (oracle entities). Underlined entities are missed by the CTRLSum (He et al., 2022) while AutoTemplate can incorporate them into the generated summary. Full version of this Table is in Appendix.

Constraint Entities: { Game Boy , Apple , Chris Gallizzi , Nintendo }
AutoTemplate: Case adds iconic Game Boy buttons to Apple handset. It also lets gamers play their existing cartridges on their handset. Developer Chris Gallizzi said: 'We wanted to create a retro device that can be easily adapted into any modern gamer's arsenal of devices' Nintendo advised keeping cartridges away from dust, where possible, to avoid gameplay glitches.
Constraint Entities: { Hyperkin , Nintendo , Game Boy Color , Start and Select }
AutoTemplate: Hyperkin has designed a case that adds the iconic directional arrows from the Nintendo Game Boy Color . It was originally devised as part of an April Fool's joke, but the popularity and demand for a real product was so high the firm has announced plans to sell it. It will feature an eight-way D-pad, two action buttons, a Start and Select button, and a battery that can be charged through the phone.

Table 8: Examples of controlled summary generation by changing constraint entities. By conditioning with different entities, the model can generate summaries with different points of interest for the same source article.

lexicalization tasks. The template generation task aims to produce unnatural text with placeholders, leading to concerns that the final output text will be less fluent than the directly generating natural text.

To this end, we compare the fluency of the output text by AutoTemplate and baselines. We specifically used the grammatical acceptability classifier based on `roberta-large` fine-tuned on CoLA dataset (Warstadt et al., 2019) following Krishna et al. (2020)[8] and show the micro averaged accuracy of sentence-level grammaticality.[9]

We show the results in Table 10. For the keywords-to-sentence generation task, AutoTemplate shows better fluency scores than the CBART model, characterizing the differences between CBART and AutoTemplate. While CBART relies on the non-autoregressive models, which leads to non-fluent text generation, AutoTemplate can be implemented on top of autoregressive models. Thus, AutoTemplate can generate more fluent output text.

For the entity-guided summarization task, AutoTemplate shows similar fluency with the state-of-the-art autoregressive text generation models, including BART and CTRLSum, indicating that the AutoTemplate can generate as fluent text as the state-of-the-art direct generation models.

Importance of Pre-training To evaluate the importance of T5 pre-training for AutoTemplate, we performed ablation studies using a *randomly* initialized model. As shown in Table 9, we confirmed that the model with pre-training significantly improves the quality of generated text in both keywords-to-sentence generation and entity-guided summarization cases. Note that the keywords-to-sentence generation model with random initialization generally produced better text quality than the baseline model, CBART, confirming the importance of using autoregressive models.

Are unique placeholders needed? Throughout this study, we assumed the unique placeholder tokens according to the order of appearance, i.e., `<X>`, `<Y>` and `<Z>`, so we investigate the importance of this design choice. We show the performance of AutoTemplate with a single type of placeholder token (i.e., `<X>` for all placeholders in the

[8] https://huggingface.co/cointegrated/roberta-large-cola-krishna2020

[9] Although we can also measure fluency using the perplexity of an external language model, it can assign low perplexity to unnatural texts containing common words (Mir et al., 2019). Therefore, we decided to evaluate fluency using the classifier.

	Keywords-to-Sentence Generation										Entity-guided Summarization							
	One-Billion-Word					Yelp					CNNDM				XSum			
	B2	B4	N2	N4	M	B2	B4	N2	N4	M	R1	R2	RL	BS	R1	R2	RL	BS
AutoTemplate	18.3	7.6	3.39	3.45	16.0	23.7	10.8	3.62	3.76	17.8	51.02	27.59	47.85	0.441	50.49	28.19	43.89	0.591
w/ random init	17.0	6.5	3.23	3.27	15.6	22.4	9.8	3.42	3.54	17.6	38.38	11.91	35.06	0.210	39.51	15.84	32.07	0.412
w/ single mask	16.6	5.9	3.15	3.19	15.0	15.9	5.2	2.86	2.92	13.8	48.05	24.53	44.69	0.387	45.67	23.07	39.31	0.493

Table 9: Ablation studies for keywords-to-sentence generation and entity-guided summarization tasks using T5-base checkpoints. B2/4 denotes BLEU-2/4, N2/4 denotes NIST-2/4, M denotes METEOR-v1.5, R1/2/L denotes ROUGE-1/2/L, and BS denotes BERTScore.

Fluency (%)	Keywords-to-Sentence	
	One-billion-words	Yelp
CBART (He, 2021)	94.42	93.95
InstructGPT (Ouyang et al., 2022)	96.57	96.94
AutoTemplate	97.05	98.15
Reference	97.25	90.77
Fluency (%)	Entity-guided summarization	
	CNNDM	XSum
BART (Lewis et al., 2020)	96.77	98.88
CTRLSum (He et al., 2022)	96.68	99.01
AutoTemplate	96.38	98.91
Reference	91.55	98.73

Table 10: Results of fluency evaluations by the acceptability classifier trained on CoLA dataset (Warstadt et al., 2019).

template $\tilde{y}$) in Table 9. We observed a significant drop in the quality of the generated text for both keywords-to-sentence generation and entity-guided summarization tasks, suggesting the importance of using unique placeholder tokens in the template.

5 Further Related Work

Template-based Text Generation For classical text generation systems, templates were an important building block (Kukich, 1983; Tanaka-Ishii et al., 1998; Reiter and Dale, 2000; Angeli et al., 2010). The advantage of a template-based system is that it can produce faithful text, but it can produce disfluent text if an inappropriate template is selected. Therefore, the current primary approach is to produce fluent text directly from the input using end-to-end neural generation models.

More recent studies have focused mainly on using templates as an auxiliary signal to control the stylistic properties of the output text, such as deriving templates as latent variables (Wiseman et al., 2018; Li and Rush, 2020; Fu et al., 2020) and using retrieved exemplars as soft templates (Cao et al., 2018; Peng et al., 2019; Hossain et al., 2020).

Copy mechanism The copy mechanism was originally introduced to deal with the out-of-vocabulary problem in machine translation by selecting the words from the source for the generation in addition to the vocabulary, such as the unknown word replacement with post-processing (Jean et al., 2015; Luong et al., 2015), and the joint modeling of unknown word probabilities into encoder-decoder models (Gu et al., 2016; Gulcehre et al., 2016), but with the advent of subword units (Sennrich et al., 2016; Kudo, 2018), the unknown word problem has been diminished. Thus, the copy mechanism is not widely used now for handling out-of-vocabulary problems.

However, the copy mechanism still plays a vital role in more complex text generation tasks such as involving numerical computation (Murakami et al., 2017; Suadaa et al., 2021) or logical reasoning (Chen et al., 2020). Specifically, they produce special tokens that serve as placeholders and replace them with the desired words in post-processing. AutoTemplate adapts a similar copy mechanism to perform lexically constrained text generation, showing that it can cover all the constrained entities in its outputs, even for more complex conditioning (more than ten entities).

6 Conclusions

This study proposes AutoTemplate, a simple yet effective framework for lexically constrained text generation. The core idea is to decompose lexically constrained text generation into two steps, template generation, and lexicalization, by converting the input and output formats. The template generation can be done with standard encoder-decoder models with beam search so that AutoTemplate can perform lexically constrained text generation without using dedicated decoding algorithms such as non-autoregressive decoding and constrained beam search. Experimental results show that the AutoTemplate significantly outperforms the competitive baselines across keywords-to-sentence generation and entity-guided summarization tasks while satisfying the lexical constraints.

7 Ethical Considerations

We do not see any ethical issues, but we would like to mention some limitations. This study proposes a method to perform hard lexically constrained text generation and shows that our proposed method could generate high-quality text in terms of the automatic evaluation metrics while satisfying the lexical constraints, but this does not guarantee the faithfulness of generated text. For example, in the summarization task, our method does not directly generate entities prone to errors, so the risk of generating summaries with unfaithful entities to the input text could be lower than existing methods. Still, the risk of generating unfaithful text in other areas remains.

References

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 502–512, Cambridge, MA. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 152–161, Melbourne, Australia. Association for Computational Linguistics.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, T. Brants, Phillip Todd Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH*.

Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. 2020. Logical natural language generation from open-domain tables. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7929–7942, Online. Association for Computational Linguistics.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380, Baltimore, Maryland, USA. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*, HLT '02, page 138–145, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. 2019. Show your work: Improved reporting of experimental results. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2185–2194, Hong Kong, China. Association for Computational Linguistics.

Angela Fan, David Grangier, and Michael Auli. 2018. Controllable abstractive summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 45–54, Melbourne, Australia. Association for Computational Linguistics.

Yao Fu, Chuanqi Tan, Bin Bi, Mosha Chen, Yansong Feng, and Alexander Rush. 2020. Latent template induction with gumbel-crfs. In *Advances in Neural Information Processing Systems*, volume 33, pages 20259–20271. Curran Associates, Inc.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 140–149, Berlin, Germany. Association for Computational Linguistics.

Junxian He, Wojciech Kryscinski, Bryan McCann, Nazneen Rajani, and Caiming Xiong. 2022. CTRLsum: Towards generic controllable text summarization. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5879–5915, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Xingwei He. 2021. Parallel refinements for lexically constrained text generation with BART. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8653–8666, Online and Punta Cana, Dominican Republic. 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*, volume 28. Curran Associates, Inc.

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546, Vancouver, Canada. Association for Computational Linguistics.

Nabil Hossain, Marjan Ghazvininejad, and Luke Zettlemoyer. 2020. Simple and effective retrieve-edit-rerank text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2532–2538, Online. Association for Computational Linguistics.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China. Association for Computational Linguistics.

Kalpesh Krishna, John Wieting, and Mohit Iyyer. 2020. Reformulating unsupervised style transfer as paraphrase generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 737–762, Online. Association for Computational Linguistics.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *21st Annual Meeting of the Association for Computational Linguistics*, pages 145–150, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Xiang Lisa Li and Alexander Rush. 2020. Posterior control of blackbox generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2731–2743, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Edward Loper and Steven Bird. 2002. NLTK: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, pages 63–70, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019a. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Ilya Loshchilov and Frank Hutter. 2019b. Decoupled weight decay regularization. In *International Conference on Learning Representations, ICLR*.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China. Association for Computational Linguistics.

Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. 2019. Cgmh: Constrained sentence generation by metropolis-hastings sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6834–6842.

Remi Mir, Bjarke Felbo, Nick Obradovich, and Iyad Rahwan. 2019. Evaluating style transfer for text. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 495–504, Minneapolis, Minnesota. Association for Computational Linguistics.

Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2016. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3349–3358, Osaka, Japan. The COLING 2016 Organizing Committee.

Soichiro Murakami, Akihiko Watanabe, Akira Miyazawa, Keiichi Goshima, Toshihiko Yanase, Hiroya Takamura, and Yusuke Miyao. 2017. Learning to generate market comments from stock prices. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1374–1384, Vancouver, Canada. Association for Computational Linguistics.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.

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

Hao Peng, Ankur Parikh, Manaal Faruqui, Bhuwan Dhingra, and Dipanjan Das. 2019. Text generation with exemplar-based adaptive decoding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2555–2565, Minneapolis, Minnesota. Association for Computational Linguistics.

Matt Post and David Vilar. 2018. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1314–1324, New Orleans, Louisiana. Association for Computational Linguistics.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Studies in Natural Language Processing. Cambridge University Press.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Lya Hulliyyatus Suadaa, Hidetaka Kamigaito, Kotaro Funakoshi, Manabu Okumura, and Hiroya Takamura. 2021. Towards table-to-text generation with numerical reasoning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1451–1465, Online. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Kumiko Tanaka-Ishii, Koiti Hasida, and Itsuki Noda. 1998. Reactive content selection in the generation of real-time soccer commentary. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 2*, pages 1282–1288, Montreal, Quebec, Canada. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2018. Learning neural templates for text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3174–3187, Brussels, Belgium. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

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 *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8649–8670, Online. Association for Computational Linguistics.

A More qualitative examples

Table 11-14 show more qualitative examples of keywords-to-sentence generation task, and Table 15 shows the full set of qualitative examples of entity-guided summarization task, including BART and reference summaries.

B Additional Experimental Details

B.1 Training details

Major hyper-parameters for training models are reported in Table 16 following the "Show-You-Work" style suggested by Dodge et al. (2019).

C Experimental details of InstructGPT

We empirically evaluated the zero-shot capability of InstructGPT (Ouyang et al., 2022) for keywords-to-sentence generation task. We specifically used `text-davinci-003` checkpoint and the prompt: `"Please create a sentence that must contain the following keywords: {{', '.join(keywords)}}."` to generate sentences that includes the pre-specified keywords. To obtain deterministic output text, we use the temperature parameter 0.

Keywords: government , ability , companies , legal
Reference: Generally , the government has the ability to compel the cooperation of private companies and assure them legal immunity with a valid court order .
CBART: The government has restricted the ability of insurance companies to take legal action .
AutoTemplate: The government has the ability to force companies to comply with legal requirements, he said.

Table 11: Example generations for the keywords-to-sentence generation on One-billion-word.

Keywords: time , voters , primary , days
Reference: At the same time , he said the more he appears before voters , the better he does on primary days .
CBART: The last time , the voters were in the primary , two days before Nov .
AutoTemplate: At the same time , voters will be able to cast their ballots during the primary days , he said.

Table 12: Example generations for the keywords-to-sentence generation on One-billion-word.

D Full results of keywords-to-sentence generation

We show non-aggregated results of keywords-to-sentence generation in Table 17. The results show that the AutoTemplate consistently outperforms baseline models.

Keywords: experience , top , easily , driver
Reference: my experience with lv cans was top notch . cab was easily flagged just off the strip , the route was direct and the driver was very nice .
CBART: the whole experience was top notch , easily by the driver .
AutoTemplate: i had a great experience with this company. they were on top of everything. i was easily able to get a driver to pick me up at my hotel.

Table 13: Example generations for the keywords-to-sentence generation on Yelp.

Keywords: southern , fresh , made , friendly
Reference: absolutely , the best pizza in southern nevada ! the pizza is always fresh , made fresh as ordered . the wait staff is very friendly and effecient !
CBART: great southern food , fresh and made with friendly staff .
AutoTemplate: this is the best southern food i have ever had. everything is fresh and made to order. the staff is very friendly and helpful. i will definitely be back.

Table 14: Example generations for the keywords-to-sentence generation on Yelp.

Constrained Entities: { Amir Khan , Manny Pacquiao , Abu Dhabi , UAE , Khan , Floyd Mayweather Jr , Las Vegas , PacMan , Bob Arum , UAE , Khan , Muslim , Brit , the Money Man , PacMan , Khan , Chris Algieri , New York }
BART (Lewis et al., 2020): Amir Khan has been linked with a fight with Manny Pacquiao . The fight could take place in Abu Dhabi in November or December. Khan is preparing to fight Chris Algieri in New York next month. Pacquiao is preparing to face Floyd Mayweather on May 2 in Las Vegas .
CTRLSum (He et al., 2022): Amir Khan could face Manny Pacquiao in Abu Dhabi , UAE . Khan has been linked with a fight with Floyd Mayweather Jr in Las Vegas . The PacMan 's promoter Bob Arum is keen for a fight in the UAE .
AutoTemplate: Amir Khan could face Manny Pacquiao in Abu Dhabi , UAE . Khan is preparing to face Floyd Mayweather Jr in Las Vegas on May 2. PacMan 's vintage promoter Bob Arum has to hand a treasure trove of an offer for a fight in the UAE this November or December. Khan is a hero of the Muslim world, the Brit would be a huge attraction there. Assuming that the Money Man wins his interim bout with PacMan next month, all that would appear to stand between him and his long-awaited mega-fight is the outside chance of a re-match. Khan is set to fight Chris Algieri in New York next month.
Reference: Amir Khan could be set to face Manny Pacquiao in Abu Dhabi , UAE . Khan 's hopes of taking on Floyd Mayweather Jr in Las Vegas have faded. PacMan 's promoter Bob Arum has a mega offer for a UAE fight late in 2015. Khan is a hero of the Muslim world and his lure in the Middle East is clear. The Brit will be ringside when the Money Man fights the PacMan on May 2. Khan must first win interim bout with Chris Algieri in New York on May 29.

Table 15: Full version of the qualitative examples including BART and reference summaries in addition to CTRLSum and AutoTemplate. Constraint entities are extracted from the reference summary (oracle entities). Underlined entities are missed by the CTRLSum (He et al., 2022) while AutoTemplate can incorporate them into the generated summary.

Computing infrastructure	NVIDIA A100
Training duration	4h
Search strategy	Manual tuning
Model implementation	[MASK]
Model checkpoint	[MASK]

Hyperparameter	**Search space**	**Best assignment**
# of training steps	50,000	50,000
validation interval	5,000	5,000
batch size	32	32

initial checkpoint for small models
initial checkpoint for base models
initial checkpoint for large models
	`google/t5-v1_1-small`
	`google/t5-v1_1-base`
	`google/t5-v1_1-large`
		`google/t5-v1_1-small`
		`google/t5-v1_1-base`
		`google/t5-v1_1-large`

label-smoothing (Szegedy et al., 2016)	*choice*[0.0, 0.1]	0.1
learning rate scheduler	linear schedule with warmup	linear schedule with warmup
warmup steps	5,000	5,000
learning rate optimizer	AdamW (Loshchilov and Hutter, 2019b)	AdamW (Loshchilov and Hutter, 2019b)
AdamW β_1	0.9	0.9
AdamW β_2	0.999	0.999
learning rate	5e-5	5e-5
weight decay	*choice*[0.0, 1e-3, 1e-2]	1e-2
max grad norm	0.1	0.1
beam width for keywords-to-sentence beam width for entity-guided summarization on CNNDM beam width for entity-guided summarization on XSUM	4 8 6	4 8 6

Table 16: AutoTemplate search space and the best assignments.

# of keywords = 1	One-Billion-Word						Yelp					
	B2	**B4**	**N2**	**N4**	**M**	**SR**	**B2**	**B4**	**N2**	**N4**	**M**	**SR**
CBART (He, 2021)	3.81	0.61	0.34	0.34	6.77	100.	5.71	1.66	0.31	0.32	8.33	100.
InstructGPT (Ouyang et al., 2022)	2.49	0.32	0.24	0.24	5.93	98.4	2.39	0.31	0.18	0.18	6.34	98.5
AutoTemplate												
w/ T5-small	5.56	0.88	1.23	1.23	9.04	100.	9.80	2.46	1.65	1.68	10.84	100.
w/ T5-base	6.01	1.01	1.36	1.36	8.82	100.	9.95	2.52	1.68	1.68	10.94	100.
w/ T5-large	**6.19**	**1.16**	**1.40**	**1.40**	**8.74**	100.	**9.78**	**2.44**	**1.67**	**1.69**	**10.99**	100.
# of keywords = 2	**B2**	**B4**	**N2**	**N4**	**M**	**SR**	**B2**	**B4**	**N2**	**N4**	**M**	**SR**
CBART (He, 2021)	7.25	1.91	0.68	0.68	10.02	100.	9.67	3.14	0.74	0.76	11.75	100.
InstructGPT (Ouyang et al., 2022)	4.57	0.84	0.48	0.49	8.68	95.2	3.94	0.66	0.30	0.30	8.89	95.0
AutoTemplate												
w/ T5-small	8.23	1.77	1.72	1.73	11.49	100.	13.46	3.94	2.14	2.18	13.09	100.
w/ T5-base	9.76	2.52	2.00	2.02	11.39	100.	13.71	4.16	2.18	2.22	13.36	100.
w/ T5-large	**10.06**	**2.59**	**2.05**	**2.06**	**11.35**	100.	**13.55**	**4.04**	**2.17**	**2.21**	**13.25**	100.
# of keywords = 3	**B2**	**B4**	**N2**	**N4**	**M**	**SR**	**B2**	**B4**	**N2**	**N4**	**M**	**SR**
CBART (He, 2021)	11.68	3.84	1.26	1.27	13.30	100.	16.03	6.48	1.73	1.77	15.75	100.
InstructGPT (Ouyang et al., 2022)	7.58	1.58	0.97	0.97	11.52	92.5	6.67	1.30	0.66	0.67	11.95	92.2
AutoTemplate												
w/ T5-small	13.20	3.73	2.60	2.62	13.76	100.	19.17	7.09	2.99	3.07	15.66	100.
w/ T5-base	15.26	5.13	2.85	2.88	14.08	100.	19.82	7.81	3.05	3.15	16.20	100.
w/ T5-large	**16.05**	**5.53**	**3.00**	**3.03**	**14.26**	100.	**20.20**	**8.11**	**3.09**	**3.19**	**16.01**	100.
# of keywords = 4	**B2**	**B4**	**N2**	**N4**	**M**	**SR**	**B2**	**B4**	**N2**	**N4**	**M**	**SR**
CBART (He, 2021)	17.67	7.07	2.31	2.34	16.92	100.	22.45	10.28	3.00	3.10	19.39	100.
InstructGPT (Ouyang et al., 2022)	11.29	3.09	1.81	1.82	14.52	91.6	10.35	2.68	1.46	1.48	15.19	90.1
AutoTemplate												
w/ T5-small	19.04	6.54	3.76	3.81	16.51	100.	25.84	10.77	3.96	4.10	18.30	100.
w/ T5-base	20.92	8.05	3.97	4.02	17.19	100.	26.87	12.26	4.02	4.21	19.03	100.
w/ T5-large	**21.23**	**8.58**	**4.01**	**4.08**	**17.29**	100.	**28.04**	**12.95**	**4.20**	**4.36**	**19.25**	100.
# of keywords = 5	**B2**	**B4**	**N2**	**N4**	**M**	**SR**	**B2**	**B4**	**N2**	**N4**	**M**	**SR**
CBART (He, 2021)	23.51	10.78	3.50	3.56	20.36	100.	27.97	13.80	4.12	4.28	22.73	100.
InstructGPT (Ouyang et al., 2022)	15.32	4.46	2.86	2.88	17.43	89.9	13.97	3.92	2.41	2.44	18.05	90.9
AutoTemplate												
w/ T5-small	23.47	9.76	4.33	4.40	19.58	100.	30.43	13.87	4.78	4.97	20.92	100.
w/ T5-base	25.97	12.03	4.68	4.78	20.44	100.	32.85	16.40	4.94	5.16	22.01	100.
w/ T5-large	**26.89**	**12.74**	**4.79**	**4.89**	**20.93**	100.	**33.11**	**16.71**	**5.05**	**5.28**	**22.18**	100.
# of keywords = 6	**B2**	**B4**	**N2**	**N4**	**M**	**SR**	**B2**	**B4**	**N2**	**N4**	**M**	**SR**
CBART (He, 2021)	29.93	15.38	4.83	4.93	23.72	100.	34.50	18.56	5.35	5.59	26.33	100.
InstructGPT (Ouyang et al., 2022)	19.50	6.71	3.93	3.97	20.20	86.4	18.33	5.76	3.50	3.55	21.01	86.3
AutoTemplate												
w/ T5-small	28.69	13.79	5.00	5.10	22.87	100.	36.31	18.99	5.53	5.80	24.03	100.
w/ T5-base	31.98	17.08	5.50	5.63	23.97	100.	38.85	21.73	5.80	6.10	25.36	100.
w/ T5-large	**33.20**	**18.18**	**5.66**	**5.80**	**24.42**	100.	**39.63**	**22.60**	**5.92**	**6.24**	**25.69**	100.

Table 17: Comprehensive results of keywords-to-sentence generation on the One-Billion-Word and Yelp datasets. **Bold-faced** and underlined denote the best and second-best scores respectively. Baseline results are copied from He (2021). B2/4 denotes BLEU-2/4, N2/4 denotes NIST-2/4, M denotes METEOR-v1.5, and SR denotes the success rate of lexical constraint satisfaction.