Extract, Denoise and Enforce: Evaluating and Improving Concept Preservation for Text-to-Text Generation

Yuning Mao[1], **Wenchang Ma**[2⋆], **Deren Lei**[3⋆], **Jiawei Han**[1], **Xiang Ren**[3]
[1]University of Illinois at Urbana-Champaign [2]Tsinghua University [3]University of Southern California
[1]{yuningm2, hanj}@illinois.edu [2]mwc17@mails.tsinghua.edu.cn [3]{derenlei, xiangren}@usc.edu

Abstract

Prior studies on text-to-text generation typically assume that the model could figure out what to attend to in the input and what to include in the output via seq2seq learning, with only the parallel training data and no additional guidance. However, it remains unclear whether current models can preserve important concepts in the source input, as seq2seq learning does not have explicit focus on the concepts and commonly used evaluation metrics also treat concepts equally important as other tokens. In this paper, we present a systematic analysis that studies whether current seq2seq models, especially pre-trained language models, are good enough for preserving important input concepts and to what extent explicitly guiding generation with the concepts as lexical constraints is beneficial. We answer the above questions by conducting extensive analytical experiments on four representative text-to-text generation tasks. Based on the observations, we then propose a simple yet effective framework to automatically extract, denoise, and enforce important input concepts as lexical constraints. This new method performs comparably or better than its unconstrained counterpart on automatic metrics, demonstrates higher coverage for concept preservation, and receives better ratings in the human evaluation.[1]

1 Introduction

Text-to-text generation is an important research problem with a broad set of applications, such as dialog response generation (Dinan et al., 2019), headline generation (Gu et al., 2020), and summarization (Mao et al., 2020b). A distinct feature of text-to-text generation (vs. free-form text generation) is that it is often desired to preserve the concepts in the source input (see Fig. 1 for an illustration). On one hand, concept preservation is crucial for maintaining the factual consistency between the input and output (Maynez et al., 2020; Nan et al., 2021). On the other hand, encouraging the model to focus on important input concepts may also improve its generation quality (Yao et al., 2019; Li et al., 2020).

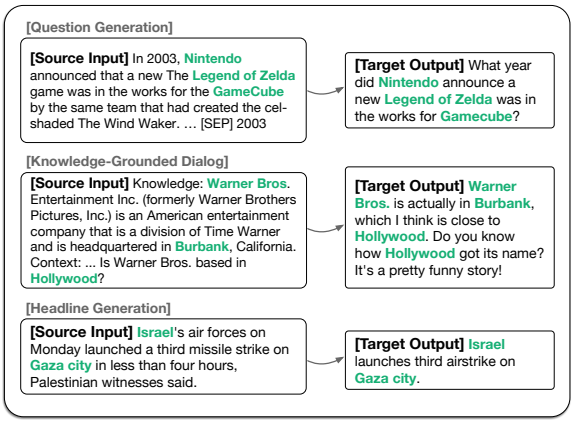

Figure 1: Examples of text-to-text generation tasks where preserving important input concepts is crucial for producing satisfactory results.

Mainstream text-to-text generation methods are mostly data-driven, which "hope" to learn meaningful mappings between source input and target output via sequence-to-sequence (seq2seq) learning – this is particularly the case for the recent pre-trained language models (PLMs) (Lewis et al., 2020; Raffel et al., 2020), where seq2seq learning is expected to identify *what to attend to* in the source input and *what to include* in the model output, with access to only parallel training data.

However, as seq2seq learning does not explicitly focus on key concepts (*e.g.*, named entities) and commonly used evaluation metrics (*e.g.*, BLEU and ROUGE) also treat all the tokens in a sequence equally important, it is unclear how many of the important input concepts can be (or have been) preserved. Existing attempts to alleviate the above

⋆Equal contribution
[1]Our code is available at https://github.com/morningmoni/EDE

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 5063–5074
November 7–11, 2021.

issue use soft constraints, such as copy mechanism or additional attention, to focus on (certain parts of) the source input (See et al., 2017; Dinan et al., 2019; Dou et al., 2021). Nevertheless, they still lack explicit guidance and resort to seq2seq learning itself to figure out what is important, without any guarantee of the model output or evaluation on the input concept preservation.

Explicit guidance for text generation can be achieved by lexically (hard) constrained generation (LCGen), which specifies lexical constraints (tokens) that must be included in the model output. However, to what extent guiding text-to-text generation with lexical constraints works in general remains unknown, as existing studies on LCGen (Hokamp and Liu, 2017; Post and Vilar, 2018; Zhang et al., 2020b) focus on scenarios where gold (ground-truth) constraints are given (*e.g.*, generate a story using user-specified keywords), while in generic text-to-text generation tasks the constraints (target output) are unavailable.

In this paper, we present a systematic analysis on generic text-to-text generation to understand (1) the abilities of seq2seq models, especially the PLMs, for preserving important input concepts and (2) whether more explicit guidance that uses important input concepts as lexical constraints can complement seq2seq learning. We select four representative tasks where preserving important concepts (entities) in the source input and incorporating them in the model output is essential, including question generation, knowledge-grounded dialog, headline generation, and abstractive summarization. We examine the effectiveness of guiding generation with the important input concepts as lexical constraints, where the concepts are either obtained by comparing the source input with the target output in an analytical study or automatically extracted from the source input in a practical setting.

Specifically, in the analytical study, we first evaluate how many of the concepts found in the target output (named *gold concepts*) are available in the source input, and how many of them are already preserved by seq2seq models. We then investigate the room for improvement if we guide generation with the gold concepts as lexical constraints (named *gold constraints*). Next, when the target output is unavailable, we propose a simple yet effective framework, named EDE, to automatically Extract, Denoise, and Enforce input concepts as lexical constraints. EDE achieves significant improvement on two tasks and moderately better or comparable performance on the other two under sequence-level automatic evaluation. Moreover, EDE receives better ratings in the human evaluation and demonstrates higher coverage in the concept-level evaluation on all the tasks that we examine.

Contributions. (1) We analyze whether current PLMs for text-to-text generation are good enough for preserving important input concepts via seq2seq learning. (2) We study the usefulness of guiding generation explicitly with important input concepts as lexical constraints. (3) We propose a framework to automatically extract, denoise, and enforce concept constraints, which achieves comparable or better performance than unconstrained generation on a range of text-to-text generation tasks.

2 Analysis on Concept Preservation

In this section, we conduct a series of analytic experiments on concept preservation, which tries to answer the following two questions:

Q1: Is the current "PLMs+seq2seq fine-tuning" paradigm for text-to-text generation good at preserving the important concepts in the source input?

Q2: What is the room for improvement if we guide text-to-text generation by enforcing high-quality (gold) concepts as lexical constraints?

2.1 Analysis Setup

We conduct extensive analytical experiments on four text-to-text generation tasks including question generation, knowledge-grounded dialog, headline generation, and abstractive summarization. All the tasks require the model to preserve important concepts (entities) in the input and include them in the output in order to achieve satisfactory results. Correspondingly, we consider the input as the source of lexical constraints and the entities in the target output as the desired (gold) constraints.

To answer Q1, we first evaluate the availability of concepts, namely how many of the gold concepts appearing in the target output can be found in the source input. We then conduct a manual analysis to study why some of the gold concepts are missing in the source input. Finally, we analyze model performance by matching gold concepts with the unconstrained model output, which reveals how many of the important input concepts are already preserved without the use of explicit constraints.

To answer Q2, we use the gold concepts as lexical constraints to guide the generation process. We

first estimate an ideal upper bound by enforcing all the gold constraints taken from the target output. We next remove the gold constraints that cannot be found in the source input and examine how large the gap is from the ideal upper bound.

2.2 Tasks and Datasets

Question Generation. Question generation is the task of generating a question given a passage and the corresponding answer. Ideally, a seq2seq model would learn to focus on the relevant information surrounding the answer span and reuse some of the concepts in the source input as part of the generated question. We use the SQuAD 1.1 dataset (Rajpurkar et al., 2016), which is repurposed by Du et al. (2017) for question generation.

Knowledge-Grounded Dialog. Knowledge-grounded dialog involves utterances with groundings to specific knowledge sources. It is used as another test case for our analysis as the model is supposed to extract important concepts from the grounded knowledge when appropriate. We use the Wizard of Wikipedia (WoW) dataset (Dinan et al., 2019) for evaluation. As we focus on generation instead of knowledge retrieval, we adopt the gold knowledge setting of WoW where the sentences with relevant knowledge are provided and use the test split that consists of new dialogues under overlapping topics with the training set.

Headline Generation. Headline generation fits our analysis as a headline usually consists of the most important concepts in the input article. We use the English Gigaword dataset (Napoles et al., 2012) for evaluation. As training on the full training set of Gigaword (Rush et al., 2015) is computationally prohibitive, we adopt two low-resource settings where the 10k training examples in Dong et al. (2019) and the first 300k of the 3.8M training examples are used.

Abstractive Summarization. Abstractive summarization is used as another testbed as a summary generally involves important concepts in the source document. We take two widely used news summarization datasets, CNN/Daily Mail (CNN/DM) (Nallapati et al., 2016) and XSum (Narayan et al., 2018), for evaluation, where CNN/DM is more extractive (Mao et al., 2020a) and XSum more abstractive (Maynez et al., 2020) in nature.

2.3 Constrained Generation

Next, we briefly introduce constrained generation and the specific methods used for our analysis.

Hard Constrained Generation. Lexically (hard) constrained generation (LCGen) specifies lexical constraints (tokens) that must be present in the model output. Compared to soft constrained generation, LCGen has the advantage of ensuring the presence of certain input concepts explicitly, but it could also be problematic when the enforced constraints are noisy (inappropriate). We choose from LCGen methods that enforce constraints by constraining beam search (Hokamp and Liu, 2017; Post and Vilar, 2018), as they function at the inference stage and can be easily combined with different seq2seq models, making our analysis more generalizable. In contrast, sampling-based and insertion-based approaches are not easily applicable to generic text-to-text generation, as they typically involve specialized training schemes or do not accept other inputs than the constraints (Miao et al., 2019; Zhang et al., 2020b; Sha, 2020).

Specifically, we take dynamic beam allocation (DBA) (Post and Vilar, 2018) for our analysis, as it has higher efficiency than other LCGen methods that constrain beam search (Hokamp and Liu, 2017). DBA revises the process of beam search by dividing the beam into a number of groups (named *banks*), each of which stores the hypotheses that satisfy the same number of constraints. DBA can ensure that every constraint is present in the model output, as the <EOS> token is only allowed when all the constraints are met. At each decoding step, there are three sources of candidates: (1) the top-k tokens across all hypotheses as in standard beam search; (2) all unfulfilled constraint tokens for each hypothesis; and (3) the single-best token for each hypothesis. The banks are trimmed by the sequence probability if the total number of candidate tokens is beyond capacity (beam size).

Soft Constrained Generation. We additionally examine the effectiveness of soft constrained generation for comparison. There are various types of soft constraints and we particularly consider the type which implicitly specifies input texts that the model needs to focus on. We use copy mechanism as a representative example for this purpose. Copy mechanism estimates the importance of tokens in the source input and learns to copy them when appropriate, which is useful for preserving the important concepts in the input, especially rare

concepts that the model does not get enough exposure to during training. For PLMs, we take the encoder-decoder cross attention in the last decoder layer as the copy distribution (Xu et al., 2020).

2.4 Experimental Settings

We use BART (Lewis et al., 2020) as the major base model for analysis. We use spaCy (Honnibal and Montani, 2017) to extract the entities from the target output as the gold concepts, the quality of which is shown to be reasonably good for the source-target alignment in summarization (Nan et al., 2021). We mainly use exact matching to be consistent with the current automatic metrics that generally consider lexical overlap, while also manually analyzing the missing concepts to address the limitation of exact matching. More implementation details can be found in App. A.

2.5 Results and Analysis

Concept Availability. We first examine how many of the gold concepts can be found in the source input. A high degree of concept overlap between the source input and target output is expected, since the tasks require groundings to the input particularly. As listed in the 3rd column of Table 1, even if the *task* requires groundings to the input, it is not uncommon that the corresponding *dataset* involves extrinsic information accidentally, *i.e.*, the target output contains information that cannot be found in the source input. The issue is most severe on Gigaword and XSum: around half of the gold concepts are not found in the source input, which coincides with recent studies on the factual consistency of text-to-text generation (Matsumaru et al., 2020; Maynez et al., 2020).

Task	$\lvert C\rvert$	$\lvert C \cap X\rvert/\lvert C\rvert$	$\lvert C \cap Y_{\text{sys}}\rvert/\lvert C\rvert$	$\lvert C \cap Y_{\text{sys}}\rvert/\lvert C \cap X\rvert$
SQuAD	1.15	73.5%	46.6%	59.4%
WoW	1.01	70.4%	57.0%	78.9%
Gigaword	1.62	**51.9%**	43.6%	75.4%
CNN/DM	**6.68**	88.7%	58.9%	66.0%
XSum	2.76	**47.7%**	48.7%	69.7%

Table 1: Statistics of concept availability (3rd column) and fulfillment (4th and 5th columns). $|C|$ denotes the number of gold concepts. X and Y_{sys} denote the concepts in source input and unconstrained model output.

Manual Analysis on Missing Concepts. To further study why certain gold concepts cannot be found in the source input, we conduct manual analysis of 50 examples on each dataset and categorize the reasons into the following groups: (Spell): the target output and source input use different terms to refer to the same concept (*e.g.*, synonyms, pronouns), which are not matched by lexical overlap. (Miss): the concept is indeed missing in the input. (NER): the concept is not a proper entity due to named entity recognition errors. (Knowledge): the concept is rephrased with commonsense knowledge (*e.g.*, "on Valentine's Day" to "in February").

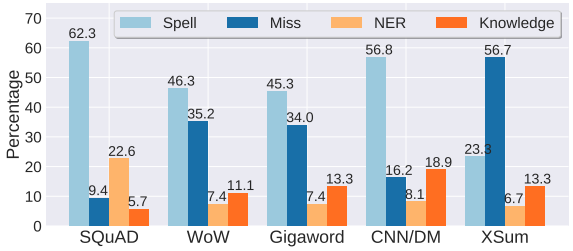

Figure 2: Categories of why certain gold concepts in the target output cannot be found in the source input.

As shown in Fig. 2, for most datasets, the major reason that gold concepts are missing is simply that they are spelled differently and not found by exact matching. That said, three of five datasets still have 10%+ missing gold concepts after ruling out this factor.[2] Such findings suggest that we need to put more effort to factual consistency when creating text-to-text generation datasets and remove problematic (hallucinated) examples from existing datasets if possible. The quality of gold concepts (NER) is reasonably high except for SQuAD. Nevertheless, after a closer look we find that most NER errors on SQuAD are repeated ones (*e.g.*, "what year" is often incorrectly recognized as *date*) that can be easily fixed manually.

Concept Fulfillment. We next study concept fulfillment, *i.e.*, matching the gold concepts with the model output to examine how many of the important concepts have been preserved without the use of constraints.

As shown in the 4th column of Table 1, roughly half of the gold concepts are preserved through standard seq2seq learning. The unconstrained model performs especially poorly on SQuAD and CNN/DM, as the gap between the concepts in the source input and those in the model output (the 3rd and 4th columns) is quite large. Interestingly, the model output contains more gold concepts than the source input on XSum, possibly because the significant level of extrinsic information in the target out-

[2] We obtain a percentage estimation of actual missing concepts by $(1 - |C \cap X|/|C|) \times \text{Miss}\%$, which is 2.5%, 10.4%, 16.4%, 1.8%, and 29.7% for the 5 datasets, respectively.

put of XSum encourages the model to hallucinate. If we only consider available gold concepts (the 5th column), the degree of concept fulfillment is improved consistently but remains relatively low.[3]

Figure 3: Room for improvement of enforcing lexical constraints when all gold constraints (All) or only those found in the input are used (Available).

Upper bounding. In Fig. 3, we list the ROUGE F1 scores when enforcing all the gold concepts as constraints (all) and only those found in the input (available). We observe that the room for improvement is usually high when all the gold constraints are used, but may become smaller if those constraints not found in the source input are excluded, especially on Gigaword and XSum, the two datasets with more extrinsic information. One exception is CNN/DM, where using available constraints performs better than using all. We hypothesize that this is because CNN/DM involves longer output and much more concepts than other tasks, which together may mislead the beam search process if the beam is mostly occupied for fulfilling the constraints. A reduced number of constraints is hence more appropriate for it.

The results above indicate that when guiding generation with important input concepts as lexical constraints, it could be seemingly difficult to improve overall output quality on datasets that involve abundant extrinsic information. That said, as commonly used metrics like ROUGE conduct sequence-level evaluation without distinguishing different tokens or identifying concepts, one also needs to examine model performance by measuring concept preservation in particular to truly reflect the effectiveness of LCGen (Sec. 4.3).

[3]The actual degree of concept fulfillment is likely to be higher as exact matching is used but the comparative analysis is still valid. More rigorously, one may again conduct manual analysis to categorize the missing concepts in model output.

3 Guiding Text-to-Text Generation with Extracted Concepts

In this section, we study a practical setting where automatically extracted concepts are used as constraints (named *automatic constraints*), with the following research question in mind:

Q3: Can we automatically extract important input concepts and use them as lexical constraints to guide text-to-text generation?

3.1 Automatic Constraint Extraction

Existing studies on LCGen generally focus on scenarios where the gold constraints are provided and the model output is centered on the constraints, but for most applications in text-to-text generation, the gold constraints (target output) are not accessible. One thus needs to conduct automatic constraint extraction from the source input. While it is not uncommon to extract keywords from the source input to help generation, previous methods (Yao et al., 2019; Li et al., 2020) typically use them as soft constraints via attention instead of lexical constraints that guide the generation process explicitly.

To extract constraints automatically, we create constraint labels on the training set by mapping gold concepts from the target output to the source input (as in the study of concept availability). We then train a state-of-the-art keyphrase extraction model, BERT-KPE (Sun et al., 2020), to predict constraints during test time. BERT-KPE uses contextualized representation and jointly optimizes keyphrase identification and ranking. We conduct constraint extraction independently as our preliminary experiments suggest that multi-task learning of text-to-text generation and constraint extraction does not appear particularly helpful. Also, decoupling the two tasks makes our framework generally applicable to different trained seq2seq models.

3.2 Constraint Denoising

Unlike using the gold constraints, directly enforcing all the automatic constraints is likely to worsen generation as some of the extracted constraints are inappropriate (see example in Fig. 4). Therefore, we propose a Denoised variant of DBA, named DDBA, which is designed specifically for dealing with automatic constraints that are noisy in nature. DDBA conducts step-level dynamic constraint denoising by modifying the DBA method as follows.

First, only constraints deemed appropriate at a decoding step are added to the beam instead of all

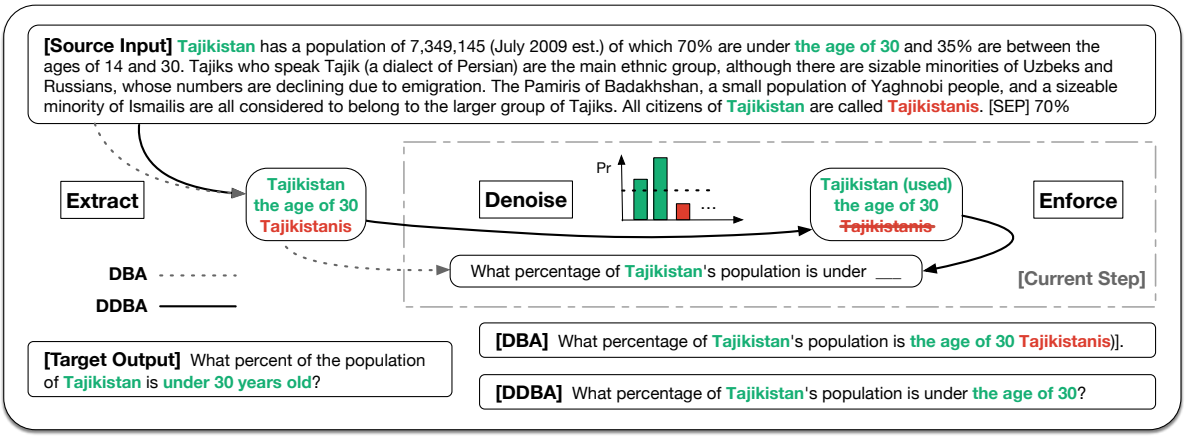

Figure 4: **Illustration of the proposed framework EDE with a real example.** DBA enforces all lexical constraints and results in nonfluency, while DDBA filters the noisy constraint and generates output with higher quality.

unmet constraints like DBA. We use the generation probability in seq2seq models to measure the appropriateness of constraints. Intuitively, inappropriate constraints that would cause nonfluency at the current step are likely to receive low probability scores and thus filtered. Another advantage of dynamic constraint denoising is that by filtering noisy constraints, more beam space is saved for better sequences that are likely to lead to a final output with higher quality. Second, unlike DBA, DDBA allows the <EOS> token even when not all the constraints are fulfilled. In this way, the model is not forced to include noisy constraints in its output. Note that not satisfying all constraints is acceptable since our goal is not to fulfill *all* the constraints but rather use them to guide generation.

Comparison w. Supervised Denoising. We considered training supervised classifiers using features such as self-attention and copy distribution to predict constraint quality, but found that simply filtering the constraints by their token probability at the current decoding step performs competitively with better efficiency, and hence use the vocabulary probability distribution as the scoring function instead (more details are in App. B). We observe that DDBA, despite its simplicity, improves the quality, especially fluency, of constrained generation under both automatic and human evaluations.

4 Experiments

In this section, we conduct experiments of guiding generation with automatic constraints and compare with the base as well as state-of-the-art models. The constraints are automatically extracted *without access to the target output*.

4.1 Automatic Evaluation

Question Generation. We show the performance comparison of question generation on SQuAD in Table 2. EDE (DBA) leads to worse BLEU-4 and ROUGE-L but higher METEOR, which is possibly due to the noise in the automatic constraints and the fact that METEOR uses stemming and synonymy matching in addition to exact matching, which successfully matches terms with different spellings. In contrast, EDE (DDBA) performs significantly better than EDE (DBA) and vanilla unconstrained BART (up to +4.3/+2.5 on BLEU-4), consistently outperforming existing baselines with $\text{BART}_{\text{base}}$ and *achieves new state-of-the-art results* with $\text{BART}_{\text{large}}$ on BLEU-4 and ROUGE-L. Copy mechanism generally leads to higher ROUGE-L but worse performance on the other two metrics.

Method	BLEU-4	METEOR	R-L
NQG (Du et al., 2017)	12.28	16.62	39.75
SemQG (Zhang and Bansal, 2019)	20.76	24.20	48.91
ERNIE-GEN$_{\text{base}}$ (Xiao et al., 2020)	22.28	25.13	50.58
BART$_{\text{base}}$	22.46	24.29	49.65
EDE (BART$_{\text{base}}$+DBA)	20.37	25.30	47.42
EDE (BART$_{\text{base}}$+DDBA)	**24.57**	**25.82**	51.73
BART-COPY$_{\text{base}}$	22.36	24.21	50.01
EDE (BART-COPY$_{\text{base}}$+DBA)	20.27	25.17	48.27
EDE (BART-COPY$_{\text{base}}$+DDBA)	24.02	25.59	**52.20**
UniLM$_{\text{large}}$ (Dong et al., 2019)	22.12	25.06	51.07
ERNIE-GEN$_{\text{large}}$ (Xiao et al., 2020)	25.40	**26.92**	52.84
BART$_{\text{large}}$	23.22	24.89	50.03
EDE (BART$_{\text{large}}$+DBA)	21.40	25.69	47.99
EDE (BART$_{\text{large}}$+DDBA)	**25.77**	26.77	52.68
BART-COPY$_{\text{large}}$	22.95	24.98	50.90
EDE (BART-COPY$_{\text{large}}$+DBA)	20.81	25.78	49.18
EDE (BART-COPY$_{\text{large}}$+DDBA)	25.00	26.61	**53.29**

Table 2: Performance comparison of question generation on SQuAD. R is short for ROUGE. **Best** and 2nd best methods are bold and underlined, respectively.

Knowledge-Grounded Dialog. We show the re-

Method	F1	R-1	R-2	R-L
ESCA-BERT (Haonan et al., 2021)	29.1	30.35	16.43	28.07
Two-Stage MemNet (Dinan et al., 2019)	30.7	31.04	16.89	28.72
E2E MemNet (Dinan et al., 2019)	35.5	37.34	19.76	33.02
$\text{BART}_{\text{base}}$	39.1	40.63	22.93	35.82
EDE ($\text{BART}_{\text{base}}$+DBA)	37.7	39.22	20.99	34.20
EDE ($\text{BART}_{\text{base}}$+DDBA)	39.5	41.25	**23.69**	**36.77**
$\text{BART-COPY}_{\text{base}}$	39.4	40.99	23.45	36.39
EDE ($\text{BART-COPY}_{\text{base}}$+DBA)	38.8	40.35	22.51	35.35
EDE ($\text{BART-COPY}_{\text{base}}$+DDBA)	**39.7**	**41.37**	23.66	36.76

Table 3: Performance comparison of knowledge-grounded dialog on WoW. R is short for ROUGE.

sults of knowledge-grounded dialog on WoW in Table 3. We observe that adding both hard and soft constraints helps with the generation process, which again implies that more explicit guidance than pure seq2seq learning could still be beneficial even for the PLMs. The improvement is consistent with the relatively high upper bound of WoW when available gold constraints are used. Also, enforcing all constraints via DBA without dynamic filtering seems to be more detrimental than helpful.

Headline Generation. We show the results of headline generation in Table 4. The improvement of EDE on Gigaword is not as much as previous tasks, which is probably because the concept availability on Gigaword is low and it is hard to extract and utilize meaningful concepts from the source input. Nevertheless, we will later show that EDE is preferred in the human evaluation and also preserves the input concepts better on Gigaword.

	Method	R-1	R-2	R-L
10k training	Transformer (Vaswani et al., 2017)	10.97	2.23	10.42
	MASS (Song et al., 2019)	25.03	9.48	23.48
	$\text{UniLM}_{\text{large}}$ (Dong et al., 2019)	32.96	14.68	30.56
	$\text{BART}_{\text{large}}$	34.08	15.45	31.28
	EDE ($\text{BART}_{\text{large}}$+DBA)	33.12	14.79	30.24
	EDE ($\text{BART}_{\text{large}}$+DDBA)	34.15	**15.56**	31.29
	$\text{BART-COPY}_{\text{large}}$	34.10	15.09	31.24
	EDE ($\text{BART-COPY}_{\text{large}}$+DBA)	33.06	14.20	30.00
	EDE ($\text{BART-COPY}_{\text{large}}$+DDBA)	**34.22**	15.23	**31.36**
300k training	$\text{BART}_{\text{base}}$	35.98	17.50	33.07
	EDE ($\text{BART}_{\text{base}}$+DBA)	34.77	16.08	31.30
	EDE ($\text{BART}_{\text{base}}$+DDBA)	36.04	17.52	33.09
	$\text{BART-COPY}_{\text{base}}$	36.35	**17.78**	33.55
	EDE ($\text{BART-COPY}_{\text{base}}$+DDBA)	35.50	15.58	32.43
	EDE ($\text{BART-COPY}_{\text{base}}$+DDBA)	**36.39**	17.72	**33.78**

Table 4: Performance comparison of headline generation on Gigaword. We only compare with BART when using 300k training examples as state-of-the-art methods use the full training set.

Abstractive Summarization. We list the results of abstractive summarization in Table 5. The performance of EDE is comparable to its unconstrained

	Method	R-1	R-2	R-L
CNN/DM	BertSum (Liu and Lapata, 2019)	42.13	19.60	39.18
	PEGASUS (Zhang et al., 2020a)	44.17	21.47	41.11
	$\text{BART}_{\text{large}}$	**44.16**	**21.28**	**40.90**
	EDE ($\text{BART}_{\text{large}}$+DBA)	43.22	20.37	40.04
	EDE ($\text{BART}_{\text{large}}$+DDBA)	44.06	20.41	40.89
XSum	BertSum (Liu and Lapata, 2019)	38.81	16.50	31.27
	PEGASUS (Zhang et al., 2020a)	47.21	24.56	39.25
	$\text{BART}_{\text{large}}$	45.14	22.27	**37.25**
	EDE ($\text{BART}_{\text{large}}$+DBA)	45.33	22.32	37.12
	EDE ($\text{BART}_{\text{large}}$+DDBA)	**45.40**	**22.38**	37.18

Table 5: Performance comparison of abstractive summarization on CNN/DM and XSum.

counterpart on both datasets. Better performance may be achieved when the constraint extraction method is improved. Apart from sequence-level evaluation (ROUGE), we observe that EDE is consistently better at concept preservation in concept-level evaluation (Sec. 4.3).

4.2 Human Evaluation

In addition to automatic evaluation, we further conduct human evaluation with the following three aspects: closeness, relevancy, and fluency (Prabhumoye et al., 2019). Closeness measures the similarity between the model output and target output. Relevancy considers the quality of model output directly with the source input, since there are usually more valid outputs than the target output for generation tasks. Fluency of the model output is on a scale of 1 (unreadable) to 4 (perfect).

We randomly sample 50 examples on each task and conduct pairwise comparisons between BART, EDE (DBA), and EDE (DDBA) with the help of three external annotators. As listed in Table 6, the results are largely consistent with automatic evaluation. For example, DBA leads to a similar number of better and worse outputs over BART on SQuAD and more worse cases on Gigaword, while DDBA outperforms BART more stably on both datasets for closeness as well as relevancy. The gaps are generally larger on relevancy than closeness, indicating that DDBA is preferred by humans when not comparing with the target output directly. DDBA also consistently outperforms DBA on both aspects and largely shares more similarities (ties) with BART due to its denoising function. The fluency ratings of BART, DBA, and DDBA are 3.82, 3.52, and 3.88 on SQuAD, and 3.34, 2.92, 3.26 on Gigaword. Such results indicate that DBA has negative impacts on generation fluency, while the fluency of

DDBA is comparable and sometimes even better than unconstrained generation.

Task	Model	Win	Tie	Lose
SQuAD Closeness	DBA vs. BART	24%	50%	26%
	DDBA vs. BART	**16%**	76%	8%
	DDBA vs. DBA	**30%**	46%	24%
Relevancy	DBA vs. BART	**18%**	66%	16%
	DDBA vs. BART	**16%**	82%	2%
	DDBA vs. DBA	**24%**	68%	8%
Gigaword Closeness	DBA vs. BART	14%	64%	**22%**
	DDBA vs. BART	**18%**	68%	14%
	DDBA vs. DBA	**24%**	72%	4%
Relevancy	DBA vs. BART	22%	50%	**28%**
	DDBA vs. BART	**26%**	54%	20%
	DDBA vs. DBA	**26%**	70%	4%

Table 6: Pairwise human evaluation comparing BART, EDE (DBA), and EDE (DDBA). We list the results on two datasets here and provide the rest in App. C.

4.3 Evaluation on Concept Preservation

As sequence-level metrics consider all the tokens equally important and can only measure overall generation quality, we further conduct concept-level evaluation to measure model performance on concept preservation in particular. We first examine the quality of extracted concepts (automatic constraints) in Table 7. We observe that the F1 scores on different datasets are largely in the range of 0.4 to 0.5, which is on par with the state-of-the-art performance on keyphrase extraction benchmarks (Meng et al., 2017).

Task	Precision	Recall	F1
SQuAD	29.1	62.9	39.8
WoW	35.4	53.1	42.5
Gigaword	41.1	47.8	44.2
CNN/DM	52.0	50.1	51.0
XSum	38.5	58.3	46.4

Table 7: Quality of automatically extracted concepts (constraints) on different text-to-text generation tasks.

Similarly, we next analyze concept preservation of seq2seq models and show the results in Fig. 5. We only consider gold concepts available in the source input for a fair evaluation. We observe that DBA generally leads to higher recall but lower precision, while DDBA balances the two metrics and consistently achieves the best F1. The improvements are most remarkable on datasets where the unconstrained model could not preserve input concepts well (*e.g.*, +3.6 on SQuAD and +1.1 on CNN/DM) and less significant if the unconstrained model already preserves most of the concepts or the dataset involves abundant extrinsic information.

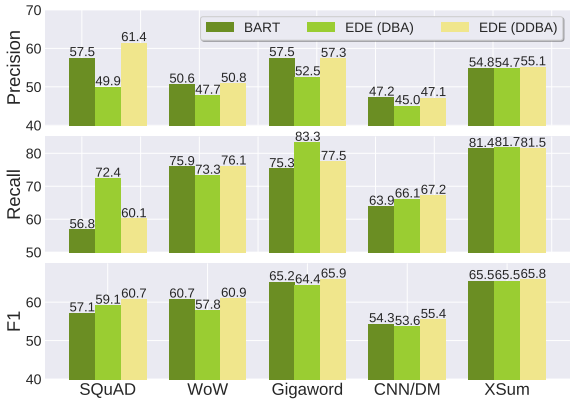

Figure 5: Comparison on concept preservation. EDE (DDBA) consistently achieves the best F1 score across different tasks.

4.4 Takeaways

From the analysis on concept preservation and model performance when automatic constraints are used, we can see that EDE is likely to improve *overall* generation quality when the upper bound performance using available gold concepts is high. EDE may not be very effective when many gold concepts cannot be found from the source input, which is an issue caused by mixed factors: (Dataset) the target output itself could be problematic and involve extrinsic concepts; (Model) concept extraction uses exact matching and misses some of the concepts; (Evaluation) existing metrics also use exact matching and would not recognize model improvement even if the concepts with different spellings are extracted. That said, EDE still achieves comparable or better performance than its unconstrained counterpart under sequence-level evaluation (overall quality) and better preserves input concepts under concept-level evaluation (concept preservation).

5 Related Work

Hard Constrained Generation. Lexically (hard) constrained generation (LCGen) has been adopted in various applications. One line of LCGen methods involves specialized model design or training schemes, which is thus not easily applicable to generic text-to-text generation (Miao et al., 2019; Zhang et al., 2020b; Sha, 2020). Other methods constrain the search space during decoding and can

be plugged into fine-tuned models without additional training (Hokamp and Liu, 2017; Post and Vilar, 2018). The constraints used in existing studies are either from external sources (Hokamp and Liu, 2017) or taken directly from the target output (Post and Vilar, 2018; Zhang et al., 2020b; Sha, 2020). Different from the previous settings of LCGen where the goal is to generate outputs satisfying the specified constraints, our study focuses on how to use lexical constraints to guide generation in more generic tasks. That is, the model can decide to take (or ignore) any constraints as long as the decisions are beneficial for generation.

Soft Constrained Generation. Soft constrained generation does not specify lexical constraints explicitly but encourages the model to attend to certain input texts. In text-to-text generation tasks where incorporating important concepts of the source input into the model output is essential, the soft constraints are usually achieved by attention mechanism to the source document (See et al., 2017; Dou et al., 2021) or some additional inputs such as keywords (Yao et al., 2019; Li et al., 2020) and external knowledge (Dinan et al., 2019).

Factual Consistency of Text-to-Text Generation. Accurately preserving the important concepts of the source input is critical for many text-to-text generation tasks to ensure the factual consistency between the source input and model output. However, recent studies (Maynez et al., 2020; Zhou et al., 2020) show that models learned in a seq2seq manner are prone to hallucinate unfaithful or nonfactual information, hindering their applicability to real-world applications. Guiding text-to-text generation with explicit constraints has been recently shown to help alleviate model hallucination and improve factual consistency (Mao et al., 2020c).

6 Conclusion

In this work, we examine whether current pretrained language models for text-to-text generation are good enough for preserving important concepts in the source input without explicit guidance but pure seq2seq learning. We conduct extensive analytical experiments on a range of text-to-text generation tasks and study when adding important input concepts as lexical constraints can help guide text-to-text generation. We propose a simple yet effective framework for automatic constraint extraction, denoising, and enforcement, which is shown to perform comparably or better than unconstrained generation in various text-to-text generation tasks and better preserves important input concepts.

Acknowledgements

We thank Yiqing Xie and anonymous reviewers for helpful feedback. Research was supported in part by US DARPA KAIROS Program No. FA8750-19-2-1004 and SocialSim Program No. W911NF-17-C-0099, National Science Foundation IIS-19-56151, IIS-17-41317, and IIS 17-04532, and the Molecule Maker Lab Institute: an AI Research Institutes program supported by NSF under Award No. 2019897. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and do not necessarily represent the views, either expressed or implied, of DARPA or the U.S. Government.

References

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019. Wizard of wikipedia: Knowledge-powered conversational agents. In *ICLR*.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *arXiv preprint arXiv:1905.03197*.

Zi-Yi Dou, Pengfei Liu, Hiroaki Hayashi, Zhengbao Jiang, and Graham Neubig. 2021. Gsum: A general framework for guided neural abstractive summarization. In *NAACL*.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1342–1352, Vancouver, Canada. Association for Computational Linguistics.

Xiaotao Gu, Yuning Mao, Jiawei Han, Jialu Liu, You Wu, Cong Yu, Daniel Finnie, Hongkun Yu, Jiaqi Zhai, and Nicholas Zukoski. 2020. Generating representative headlines for news stories. In *Proceedings of The Web Conference 2020*, pages 1773–1784.

Wang Haonan, Gao Yang, Bai Yu, Mirella Lapata, and Huang Heyan. 2021. Exploring explainable selection to control abstractive generation. *AAAI*.

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.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

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.

Haoran Li, Junnan Zhu, Jiajun Zhang, Chengqing Zong, and Xiaodong He. 2020. Keywords-guided abstractive sentence summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8196–8203.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. 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 3730–3740, Hong Kong, China. Association for Computational Linguistics.

Yuning Mao, Liyuan Liu, Qi Zhu, Xiang Ren, and Jiawei Han. 2020a. Facet-aware evaluation for extractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4941–4957, Online. Association for Computational Linguistics.

Yuning Mao, Yanru Qu, Yiqing Xie, Xiang Ren, and Jiawei Han. 2020b. Multi-document summarization with maximal marginal relevance-guided reinforcement learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1737–1751, Online. Association for Computational Linguistics.

Yuning Mao, Xiang Ren, Heng Ji, and Jiawei Han. 2020c. Constrained abstractive summarization: Preserving factual consistency with constrained generation. *arXiv preprint arXiv:2010.12723*.

Kazuki Matsumaru, Sho Takase, and Naoaki Okazaki. 2020. Improving truthfulness of headline generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1335–1346, Online. Association for Computational Linguistics.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online. Association for Computational Linguistics.

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 582–592, Vancouver, Canada. 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.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 280–290. ACL.

Feng Nan, Ramesh Nallapati, Zhiguo Wang, Cicero Nogueira dos Santos, Henghui Zhu, Dejiao Zhang, Kathleen McKeown, and Bing Xiang. 2021. Entity-level factual consistency of abstractive text summarization. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2727–2733, Online. Association for Computational Linguistics.

Courtney Napoles, Matthew R Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX)*, pages 95–100.

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.

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.

Shrimai Prabhumoye, Chris Quirk, and Michel Galley. 2019. Towards content transfer through grounded text generation. 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 2622–2632, Minneapolis, Minnesota. Association for Computational Linguistics.

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*, 21:1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Lei Sha. 2020. Gradient-guided unsupervised lexically constrained text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8692–8703, Online. Association for Computational Linguistics.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. In *International Conference on Machine Learning*, pages 5926–5936. PMLR.

Si Sun, Chenyan Xiong, Zhenghao Liu, Zhiyuan Liu, and Jie Bao. 2020. Joint keyphrase chunking and salience ranking with bert. *arXiv preprint arXiv:2004.13639*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010.

Dongling Xiao, Han Zhang, Yukun Li, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2020. Ernie-gen: An enhanced multi-flow pre-training and fine-tuning framework for natural language generation. *arXiv preprint arXiv:2001.11314*.

Song Xu, Haoran Li, Peng Yuan, Youzheng Wu, Xiaodong He, and Bowen Zhou. 2020. Self-attention guided copy mechanism for abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1355–1362, Online. Association for Computational Linguistics.

Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7378–7385.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J Liu. 2020a. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *ICML*.

Shiyue Zhang and Mohit Bansal. 2019. Addressing semantic drift in question generation for semi-supervised question answering. 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 2495–2509, Hong Kong, China. Association for Computational Linguistics.

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.

Chunting Zhou, Graham Neubig, Jiatao Gu, Mona Diab, Paco Guzman, Luke Zettlemoyer, and Marjan Ghazvininejad. 2020. Detecting hallucinated content in conditional neural sequence generation. *arXiv preprint arXiv:2011.02593*.

A Implementation Details

We use Nvidia GTX 2080 Ti and Quadro RTX 8000 for the training of $\text{BART}_{\text{base}}$ and $\text{BART}_{\text{large}}$, respectively. The evaluation metrics for each task are consistent with existing methods for comparison. For constraint extraction, we remove a constraint completely before entering the generation process if its score predicted by the keyphrase extraction model is lower than a threshold (tuned to optimize constraint F1 score for each task). LCGen is typically slower than standard beam search but the runtime overhead is acceptable when the beam size is not too large (we set it no larger than 20) and the number of constraints is small (most tasks involve one to two constraints on average).

B Supervised Denoising

In our preliminary experiments, we considered training a supervised classifier to estimate step-level constraint importance during decoding. Specifically, we use the following features to measure constraint importance: two dynamic features based on the token probability in the vocabulary distribution and the token probability in the copy distribution (if available); Two static features, out-degree and in-degree centrality defined below, based on the transformer self-attention of source tokens.

Source token centrality is based on a directed graph built upon the self-attention (Xu et al., 2020). Let $G = (V, E)$ denote a graph representing the encoder self-attention, where vertices V denote the source tokens and $E(i, j)$ denote the self-attention from token i to j with $\sum_i E(i, j) = 1$. The out-degree centrality of token i is defined as $\sum_j E(i, j)$, which measures the degree a token i contributes to other tokens. A transition probability matrix T is defined as $T(i, j) = E(i, j)/\sum_j E(i, j)$, where the self-attention is normalized by the out-degree centrality. The in-degree centrality of token i is defined as $\sum_j T(j, i)$.

We first run EDE and store the intermediate dynamic features during decoding. Then, those gold constraints with a generation probability greater than a threshold at a certain decoding step are treated as positive examples. As the features are all scalar, we use random forest as the classifier due to efficiency considerations. Other conventional classifiers performed worse in our experiments. The comparison between DDBA (supervised) and DDBA (unsupervised) is shown in Table 8. We observe that the performance of the two variants is rather similar, which is possibly because there are no step-level labels that determine whether it is appropriate to use a constraint at a specific decoding step, and we have to use approximate labels that also depend on the generation probability.

We also explored using reinforcement learning that assigns sequence-level reward (*e.g.*, the scores on evaluation metrics) to bypass the lack of step-level labels. However, reinforcement learning turned out to be unstable and costly to learn.

Method	BLEU-4	METEOR	R-L
DDBA (supervised)	24.14	25.43	51.67
DDBA (unsupervised)	23.98	25.44	51.69

Table 8: Comparison of DDBA (supervised) and DDBA (unsupervised) on question generation.

C Human Evaluation

In Table 9, we provide additional results on human evaluation. The results are largely consistent with automatic evaluation: EDE (DDBA) performs significantly better than BART on question and dialog generation, moderately better on headline generation, and comparable on summarization.

Task	Model	Win	Tie	Lose
WoW Closeness	DBA vs. BART	12%	60%	28%
	DDBA vs. BART	18%	70%	12%
	DDBA vs. DBA	32%	52%	16%
Relevancy	DBA vs. BART	14%	62%	24%
	DDBA vs. BART	22%	64%	14%
	DDBA vs. DBA	34%	54%	12%
CNN/DM Closeness	DBA vs. BART	16%	58%	26%
	DDBA vs. BART	12%	78%	10%
	DDBA vs. DBA	24%	70%	6%
Relevancy	DBA vs. BART	26%	50%	24%
	DDBA vs. BART	16%	70%	14%
	DDBA vs. DBA	20%	68%	12%
XSum Closeness	DBA vs. BART	14%	78%	8%
	DDBA vs. BART	16%	78%	6%
	DDBA vs. DBA	20%	70%	10%
Relevancy	DBA vs. BART	10%	78%	12%
	DDBA vs. BART	16%	70%	14%
	DDBA vs. DBA	16%	76%	8%

Table 9: Pairwise human evaluation comparing the base model BART, EDE (DBA), and EDE (DDBA).