Controllable Paraphrase Generation with Multiple Types of Constraints

Nazanin Dehghani
Univ Rennes, CNRS, IRISA
Lannion, France
nazanin.dehghani@irisa.fr

Hassan Hajipoor
Univ Rennes, CNRS, IRISA
Lannion, France
hassan.hajipoor@irisa.fr

Jonathan Chevelu*
Univ Rennes, CNRS, IRISA
Lannion, France
jonathan.chevelu@irisa.fr

Gwénolé Lecorvé
Orange / Univ Rennes, CNRS, IRISA
Lannion, France
gwenole.lecorve@orange.com

Abstract

One of the current challenges in paraphrase generation is the ability to enforce linguistic constraints on the desired output. These constraints may relate to the length of the output sentence, its syntactic structure, the presence of specific words, etc. While recent works focus on these constraints in isolation, this paper studies a variety of constraints imposed separately or in combination with one another. These constraints include different linguistic factors (surface words, syntax and semantics) of the input sequence or output sequence or both, and different data shapes (scalar value, sequence and tree). These constraints are integrated in a paraphrase generation process using an attention-based encoder-decoder model trained and experimented on the ParaNMT-50M corpus. The results show that the constraints are well respected by the models and that they allow to improve the quality of the produced paraphrases. This multiple constraint-driven model opens a new window for controllable paraphrase generation. The code is publicly available[2].

1 Introduction

Paraphrase generation, *i.e.* the transformation of a sentence into a well-formed but lexically different one while preserving its original meaning, is a fundamental task of Natural Language Processing (NLP). Through the objective of meaning preservation, paraphrase generation systems are useful tools when users attempt to rephrase some parts of a text [Max, 2008]. It can also be applied to improve other NLP tasks, for instance by increasing the training data with new examples of sentences [Iyyer et al., 2018] or by offering variation to the outputs of a given model. Examples of such application tasks are text simplification [Kajiwara, 2019] or speech synthesis [Boidin et al., 2009].

In most applications, the objective of paraphrase generation is to fit a target domain/task/style based on target-specific samples (e.g., 'Twitter', 'Shakespeare', etc.). For instance, Keskar et al. [2019] have proposed an effective solution to perform style transfer using the concept of style tokens or "codes" to condition transformer models. As opposed to these approaches, we focus on applications where the user wants precise control on the generation process by selecting the linguistic parameters that best fit its needs. To do so, related work have proposed to enforce constraints on the paraphrase's

*Corresponding author

[2] https://gitlab.inria.fr/expression/tremolo/controllable-paraphrase-generation

1st CtrlGen: Controllable Generative Modeling in Language and Vision Workshop at NeurIPS 2021.

length [Kikuchi et al., 2016], its syntactic structure [Kumar et al., 2020, Goyal and Durrett, 2020], or words to be removed from the source sentence [Kajiwara, 2019].

In this context, this paper studies to what extent the constraint-sensitive paraphrase generation paradigm can be extended. Namely, our contribution is to consider, compare, and combine several types of constraints, depicting (1) various linguistic aspects (syntax, lexicon and semantics), (2) focusing either on the output or input sentence or both, and (3) with different data shapes (scalar value, sequence, tree). To do so, multiple attention-based encoder-decoder models[3] are trained following a unique consistent architecture. Based on a general-purpose paraphrase corpus (ParaNMT-50M), constraints are derived from each source-target pair of sentences and used to condition the paraphrase model. The performance of the models is investigated across the different types of constraints and their combinations through objective and perceptual evaluations of the paraphrases, as well as by measuring how often the input constraints are satisfied. Finally, it is important to highlight that the generation of the constraints is not the purpose of this paper. In absolute terms, they could be provided by end-users or predicted by some machine learning models. Instead, this paper focuses on how well the single or combined constraints can be satisfied using the paraphrase model.

The paper is organized as follows: after discussing the related work in Section 2, Section 3 details our generic model architecture while Section 4 introduces the different types of constraints tested. Then, Section 5 presents the experimental setup, and Sections 6 and 7 report the results.

2 Related work

While natural language generation was historically grounded on rule/grammar-based systems [Deemter et al., 2005, Gatt and Reiter, 2009] or statistical systems with dedicated modules [Quirk et al., 2004, Koehn, 2009], advances in deep learning have homogenized the various tasks (machine translation, paraphrase generation, text summarization, question-answering, etc.) with a massive use of sequence-to-sequence neural models, whatever the underlying implementation (RNNs, LSTMs, Transformers, etc.). This section focuses on these approaches but note that other paradigms still exist, for instance the one based on retrieval approaches [Li et al., 2018, Huang et al., 2019].

In the case of paraphrase generation, this homogenisation allows the translation of the key dimensions of the task, i.e. semantic preservation, linguistic correctness and respect of the intended task ("what is the paraphrase for?"). The topic of *controllable paraphrase generation* studies how can this last dimension best be taken into account. When the purpose is to mimic a target style, this topic can be seen as *style transfer* [Jin et al., 2020]. Both topics have received much attention in recent years.

The first (straightforward) approach is to leave the model infer the logic of the task-specific goal by simply observing the data. If aligned data is available, the output distribution of the model can be conditioned by learning the transition from input sentences to a paraphrase which satisfies the target task [Wang et al., 2019, Lai et al., 2021]. Other methods explicit inside the model the need to separate various dimensions/concepts from the input sentence. Then, the problem is either to keep certain aspects invariant (eg. semantics and formality level) in the output while changing the others (eg. syntax) [Bao et al., 2019, John et al., 2019], or to modify all aspects independently and recombine them [Li et al., 2019]. An interesting outcome of these approaches is that concepts (embeddings) from several input sentences can be mixed together (eg. meaning of sentence A with syntax of sentence B). Another interesting perspective of this approach is that it can be implemented in an unsupervised context (i.e. where the input and output sentences are not aligned) thanks to adversarial training. In general, the negative point of these approaches for our problem is that the control is not precise because the notion of style is not explicit.

For a finer control of the generated paraphrases, other approaches have been proposed to integrate explicit constraints. For instance, related work have proposed to enforce constraints on the paraphrase's length [Kikuchi et al., 2016], its syntactic structure [Kumar et al., 2020, Goyal and Durrett, 2020], words to be removed from the source sentence [Kajiwara, 2019] or even the name of a target style [Dai et al., 2019]. In the line with these works, our article experiments in depth the capacity of this approach to integrate multiple and varied types of constraints. This principle can also be found in approaches that rely on prompts [Liu et al., 2021]. The idea is to give indications to the

[3] As [Egonmwan and Chali, 2019] have shown that the difference between transformer and recurrent neural networks (RNNs) is not so big in paraphrase generation, we use RNNs to be able to compare with related works.

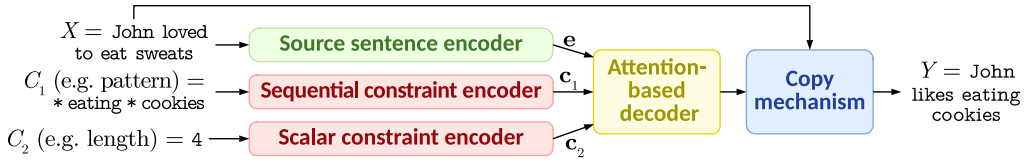


Figure 1: Overview of the constraint-sensitive model.

models as to the expected output. For example, this can be done using prefix prompts with global instructions ("paraphrase like Shakespeare: your sentence...") [Keskar et al., 2019] or by masked output sentences in which unmasked words could be thought as known constraints [Brown et al., 2020]. Alternatively, the constraints can be taken into account at the time of decoding [Post and Vilar, 2018, Hu et al., 2019]. These approaches have the advantage of not requiring a particular generation model a priori, but this requires efficient decoding strategies to face the combinatory of the search space [Chevelu et al., 2009, Fabre et al., 2021]. Furthermore, they do not guarantee that the model proposes paraphrases that satisfy the constraints. In this respect, variational [Deriu and Cieliebak, 2018] or GAN [Cao and Wan, 2020] can help bringing more diversity.

3 Constraint-sensitive model

Given an input sentence $X = [x_1, ..., x_{T_X}]$ and a set of constraints $\mathcal{C} = \{C_i\}_{1 \leq i \leq n}$ of target characteristics, our goal is to generate a paraphrase sentence $Y = [y_1, ..., y_{T_Y}]$ that satisfies the constraints. Each constraint can be either a single scalar value, i.e., the desired length of output, or can be complex/structured data, i.e., a pattern to insert, a syntax tree, etc. In the paper, we refer to this architecture as Constraint-sensitive Paraphrase Generation Network (CPGN).

Inspired by Iyyer et al. [2018], we extend the usual encoder-decoder architecture to include several encoders, one for the source sequence and one for each constraint to apply, as shown in Figure 1. For input sentences and sequential constraints, each encoder relies on an embedding layer, then a bidirectional LSTM layer, while it is limited to an embedding layer for scalar constraints. The decoder relies on a unidirectional LSTM backed up by an attention mechanism for each sequential encoder. Finally, a copy mechanism mixes direct information from the source sentence and from the decoder to decide whether some source tokens should be simply duplicated. The details of this architecture can be found in the supplementary material. To study the various types of constraints and their potential combinations, several instances of this architecture are trained and compared.

4 Constraints

This section describes different types of constraints, ranging from various linguistic aspects (syntax, lexicon, semantics) and data shapes (scalar, sequence, tree) and how they are modeled in this work. Contrary to macroscopic styles (like 'Twitter' or 'Shakespeare'), we focus on linguistic constraints that users can control. For each type of constraints, examples are given from real data.

4.1 Length

As illustrated in the example below, the output sentence can be constrained based on a target length, i.e., number of tokens, of the output sentence. This is directly represented as a single integer which is then mapped by the dedicated encoder to an embedding of size 56.

Input	Constraint	Output
well , we 're going to make sure the tape is in the right hands	length = *10*	*we 'll make sure the tape is in the hands*

4.2 Syntactic structure

The syntactic structure of a sentence can be expressed as a parse tree where the internal nodes are phrases and leaves are tokens. To constrain a paraphrase to follow a given syntactic structure, leaves are removed and the remaining tree is linearized using a prefix-bracketed representation. If desired, the syntactic constraint can be

relaxed by only keeping the few first top levels of the tree as shown in the example below. As defined in Section 3, all items of the constraint sequence are mapped to an embedding space, and then contextually embedded with a recurrent layer [4]. Here, the vocabulary of the constraint is of size 104, token embeddings are of size 56, and the contextual embeddings from the LSTM layer of size 128.

Input	Constraint	Output
you ca n't just take her life and forget about ours .	top-2 parse tree = *(ROOT (SBARQ (WHADVP) (SQ) (.)))*	*how can you take her life and forget ours ?*

4.3 Semantic similarity

Given an input sentence, the goal is to generate a paraphrase with a desired semantic similarity. For this purpose, we use BERTScore [Zhang et al., 2020] which successfully incorporates semantic information behind sentences [Devlin et al., 2019]. Although a paraphrase is generally understood as preserving as much semantics as possible from the original sentence, the idea of this type of constraint is that the user can relax this criterion to offer more flexibility to the model and possibly satisfy other types of constraints. Examples with imperfect meaning preservation can be found in real data. For instance, the following example shows a pair of paraphrases where the semantic similarity is only 80%. In practice, this similarity cannot be too low as this would break the definition of paraphrases. Hence, in our training dataset, the BERTScore values only range from 66 % to 100 %.[5]

Input	Constraint	Output
after all i saw , the idea of going back to mom and dad depresses me .	BERTScore Similarity = *80%*	*and my dad 's gon na hurt me .*

4.4 Pattern of desired words

It is common that a user may want to include specific words in the output paraphrase. For this purpose, we introduce *inclusive pattern* (pattern$_{\in}$), where a pattern of desirable words is provided to the network in the simple form of a regular expression. In the example, we ask the network to paraphrase the input in a way that the words *violated* and *ceasefire* appear in the output in this order. As shown in the example, regular expression operators can be used. Here, asterisks (*) stem for "*zero or more words*", and dollar sign (*$*) for "*end of sentence*". Likewise, patterns can be asked to start at the beginning of the sentence (not shown in the example). In our experiments, we limit patterns to include 1 or 2 words, as well as the previously mentioned operators. This enables us to represent a pattern constraint as a fixed-length sequence of tokens. Each pattern token (words and operators) is embedded using the same embedding space as in the source sequence encoder.

Input	Constraint	Output
they 've broken the armistice	patt.$_{\in}$ = * *violated * ceasefire $*	*they violated the ceasefire*

4.5 Pattern of undesired words

Alternatively, users may want to discard some words that are in the source sentence but should *not* appear in the output sentence. These constraints are named *exclusive patterns*, denoted as pattern$_{\notin}$. They are modeled and fed into a network in the same way as pattern$_{\in}$. In the following example, the input sentence is paraphrased by removing the unwanted sequence *the fuck*.

Input	Constraint	Output
get the fuck out !	pattern$_{\notin}$ = * *the fuck* *	*get out of here !*

4.6 Combination of constraints

Finally, we study the possibility for the network to jointly take into account several types of constraints. For instance, below is shown a combination of length and pattern constraints which leads to a paraphrase with length 13 in which *seeing* and *another* have been inserted (here replacing *dating* and *different*, respectively). As stated in Section 3, all constraints are independently encoded and fed to the decoder as additional input information.

[4] For the experiments, parse trees are provided by the Stanford parser [Manning et al., 2014].

[5] Let one note that a wider range could be considered by extending the corpus with poor quality paraphrases. However, we decided here to stick to the original training data to propose a consistent experimental pipeline across all the constraints.

Input	Constraint	Output
if dick starts dating her again , you 're gonna need to get a different roommate	length = *13* ***and*** pattern$_{\in}$ = * *seeing* * *another* *	*if dick starts seeing her again , you have to find another roommate*

5 Experimental setup

This paper studies the ability of CPGNs to respect the provided constraints while producing semantically and grammatically correct paraphrases. This section details the data, as well as systems, evaluation metrics and the constraint selection scheme.

5.1 Dataset

All experiments are conducted on the ParaNMT-50M corpus [Wieting and Gimpel, 2018] which includes 50 million paraphrases with a vocabulary size of $31K$ obtained by back-translating the Czech side of the CzEng parallel corpus [Bojar et al., 2016]. Among them, $49M$, $12K$ and $100K$ were used for training, validation and test, respectively.

5.2 Systems

Our Models (CPGNs) are trained separately for each single type of constraints from Section 4 and some combinations of them.

The no-constraint baseline is a model with no constraint encoder module.

Naive systems are constraint-aware systems which built by designing constraint-specific post-processing steps. These steps can be either directly applied on the input sequence or on the output sequence given by the baseline constraint-free model (*No Constraint*). In the remainder, these two configurations are referred to as *Naive* or *Baseline+Naive*, respectively. In the case of length constraints, post-processing consists in inserting or deleting random tokens until the target length is met. Regarding patterns, the presence (*pattern*$_{\in}$) or absence (*pattern*$_{\notin}$) of the regex constraint is forced by replacing random tokens with pattern ones, or inversely.

Length and syntactic baselines are models from the litterature. *LenEmb* [Kikuchi et al., 2016] proposes to control the length of output via the embedding of the remaining length and also with *LRPE* [Takase and Okazaki, 2019] that proposed to use positional encoding to control the length of output[6]. In addition, we compare with *SCPN* [Iyyer et al., 2018] that generates a paraphrase from full parse tree.

5.3 Evaluation metrics

Constraint evaluation metrics Evaluation focuses on how well the constraints are effectively satisfied in the outputs. This is computed as mean absolute errors (MAEs)

- $\text{MAE}_{length} = \frac{1}{|N|}\sum_N \left|\frac{len(\text{output})-\ell}{\ell}\right|$ where $len(x)$ is the number of tokens in sentence x, ℓ is the target length, and N is the normalization factor.
- $\text{MAE}_{\text{BERTScore}} = \frac{1}{|N|}\sum_N |\,\text{BERTScore(input, output)} - b\,|$ where b is the target meaning preservation.
- $\text{MAE}_{\text{pattern}_{\in/\notin}} = 1 - \frac{\text{\# of matched patterns}}{\text{\# of constraint patterns}}$.

Error on parse trees is reported in terms of *syntactic-TED* metric proposed by Chen et al. [2019] which is based on evaluating the tree edit distance between parse the tree of output and the desired parse tree after removing word tokens (tree leaves).

Paraphrase evaluation metrics We rely on the standard automatic NLG evaluation metric BLEU [Papineni et al., 2002]. BERTScore values between the output and reference are also reported to evaluate their semantic similarity[7].

5.4 Choice of the constraints values

Two sets of experiments are set up: first in a controlled environment where the constraints are set such that the system targets the reference. Second, without references, we explore more freely the impacts of values of the constraints.

[6]Both of these works proposed to control the length in *character* level, however, ours works in *word* level.

[7]See supplementary materials for the results in terms of METEOR, ROUGE-L and GSM.

Table 1: Paraphrase quality (BLEU and BS(o,r), the higher the better) and MAEs (the lower the better) for various types of constraints. MAEs associated with the system constraints are in bold. BS(o, r) stands for the BERTScore between the output, i.e. the generated paraphrase, and the reference. Models prefixed with $\star$ are based on the proposed CPGN architecture. "All types" includes length, top-2 Parse Tree, BERTScore, pattern$_{\in}$ and pattern$_{\notin}$.

Constr.	**Model**	**NLG metrics** ↑		**M**ean **A**bsolute **E**rror (**MAE**) (%) ↓				
		BLEU	BS(o,r)	Length	TED	BSc.	Patt.$_{\in}$	Patt.$_{\notin}$
None	$\star$No-Constraint Baseline	30.0	92.2	22.2	12.1	4.8	72.9	52.03
Length	Naive Length	23.2	90.5	**0.0**	14.3	3.2	100.0	84.5
	LenEmb [Kikuchi et al., 2016]	26.2	91.0	**12.2**	13.7	3.9	94.2	61.7
	LRPE [Takase and Okazaki, 2019]	29.4	91.6	**10.7**	11.6	3.7	59.1	46.3
	$\star$Length	31.6	92.7	$\sim$ **0.0**	11.4	3.3	66.3	49.9
Syntax	SCPN [Iyyer et al., 2018]	25.9	92.1	23.2	**10.9**	3.4	63.1	59.3
	$\star$Top-2 Parse Tree	30.7	92.7	18.7	**10.8**	4.7	60.0	48.6
	$\star$Full Parse Tree	52.9	95.5	1.7	**2.5**	3.8	38.8	33.4
Semantics	$\star$BERTScore	31.0	93.2	20.4	11.5	**3.2**	64.2	48.9
Pattern$_{\in}$	Naive Pattern$_{\in}$	22.1	90.4	24.5	19.3	4.7	**0.0**	0.7
	Baseline + Naive Pattern$_{\in}$	25.8	91.1	22.9	16.9	4.1	**0.0**	0.5
	$\star$Pattern$_{\in}$	44.7	94.8	15.5	9.3	3.2	**7.4**	35.2
Pattern$_{\notin}$	Naive Pattern$_{\notin}$	19.8	90.1	24.5	21.0	7.2	100.0	**0.0**
	Baseline + Naive Pattern$_{\notin}$	21.7	90.8	23.9	07.9	5.2	79.5	**0.0**
	$\star$Pattern$_{\notin}$	34.7	93.1	18.0	10.8	4.4	61.5	**41.2**
Multiple	$\star$Length + BERTScore	31.9	93.9	**< 0.1**	11.1	**2.6**	62.5	46.8
	$\star$Length + Top-2 ParseTree	33.0	93.2	$\sim$ **0.0**	**8.3**	2.6	57.1	42.9
	$\star$Length + Pattern$_{\in}$	46.6	95.2	$\sim$ **0.0**	8.7	3.8	**14.0**	34.8
	$\star$BERTScore + Pattern$_{\in}$	45.6	95.1	13.0	8.8	**4.0**	**21.8**	33.5
	$\star$All types	47.2	94.8	**2.7**	**8.8**	**3.8**	**11.3**	**26.6**

Reference-based experiments To automatically assess the quality of constrained paraphrase generators, a test set of paraphrase pairs is used and, for each source sentence, constraints are selected to fit the corresponding reference, as in [Takase and Okazaki, 2019]. For example, if the length of the reference is 10, the network is asked to generate a paraphrase with length 10 and the output is compared against the reference. While enabling the use of measures like BLEU or MAE, this setup introduces a bias on the paraphrase quality since the constraints can always be satisfied within a linguistically correct paraphrase (the reference).

Free constraints experiments To investigate how the system generalizes in a less controlled environment, we explore a plausible range of constraint values, regardless of the reference. Thereby, for length constraint, we test values from 5 to 40 for each input in the test set. For *BERTScore*, we consider the values in the range of 0.6 to 1.0 with step 0.05. For *pattern*$_{\in}$, we construct maximum 5 patterns by randomly choosing the synonym of input tokens[8]. For *pattern*$_{\notin}$, we construct maximum 5 patterns by randomly choosing one or two input tokens. We extracted the most 10 frequent top-level *parse trees* from the training set and generated paraphrases of each sample with these 10 parse trees. Note that for these experiments, natural language generation (NLG) metrics cannot be computed since no reference exists for each constraint. Only constraint satisfaction can be reported.

Therefore, three workers were asked to rate 150 paraphrase pairs: 25 pairs for 5 types of constraints, and 25 pairs from the no-constraint baseline. All source sentences are different and have been randomly chosen. For some types of constraints, constraint values have been randomly drawn from the most likely values to prevent from outliers [9]. Length constraint range in $[-20\%, +20\%]$ of the input sentence length ; BERTScores are in $[0.8, 0.95]$; and parse trees were limited to the 5 most frequent one (instead of 10 in MAE experiments).

6 Reference-based results

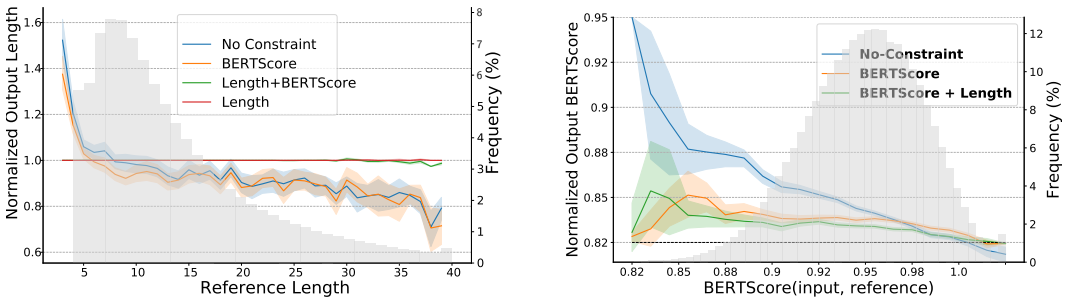
As detailed in Table 1, this section reports paraphrase quality and constraint satisfaction when targeting a given reference.

6.1 Constraint satisfaction

The following comments on the main results w.r.t. constraints satisfaction (MAE columns).

[8] We used Word2Vec [Mikolov et al., 2013] and WordNet [Miller, 1998] to extract synonyms.

[9] For instance, targeting a length of 40 words for an input sentence of 10 words.



(a) Length constraint satisfaction.

(b) BERTScore constraint satisfaction.

Figure 2: Detailed constraint satisfaction for length (a) and BERTScore (b). Lines show the ratio between requested and generated constraints, 1 meaning that they exactly match. Gray histograms are the distribution of constraints in the training set. Details in Section 6.1.

Length is precisely followed. The minimum $\text{MAE}_{\text{length}}$ is obtained when *length* is applied as a constraint and the error is significantly reduced compared to when other types of constraints are applied. The baselines *LenEmb* [Kikuchi et al., 2016] and *LRPE* [Takase and Okazaki, 2019] that also include a length constraint perform worse because they try to follow the target length at the character level, which is much harder. Then, the use of the reference full parse-tree brings very good results, which is logical as the number of leaves in this tree is basically the number of tokens. Figure 2a provides additional details regarding the requested (reference) length. The *y-axis* shows the length of the produced paraphrase normalized by the requested length. Overall, it appears that the model properly learns to generate the desired length even for long sentences. On the other hand, other systems like no-constraint or BERTScore tend to produce shorter sentences, especially when the reference is long (more than 20 tokens). This is in line with previous findings showing that neural paraphrase generation tends to produce short sentences and thus performs poorly on long ones [Prakash et al., 2016].

The more structural information, the better the network follows it. To investigate syntactic constraint satisfaction, full and top-2 level parse trees from the reference are compared. A significant improvement in TED is reported when the full parse tree is provided. Considering the top-2 levels is much less efficient, but still outperforming the no-constraint baseline. This is because the top-2 levels are too generic to describe the desired changes. As an illustration, the top-2 levels of parse trees of the input and reference are the same for 24% of the test sentences. Finally, it is worth noting that all types of constraints lead to better TED values than the *No-Constraint* baseline.

Semantics can be controlled by BERTScore representations. Results in Table 1 show that the highest semantic similarities with the reference are for the models based on the BERTScore constraints. This means that semantics can be controlled in the paraphrase generation process. Likewise, it is interesting to highlight again that BERTScore values come from a very limited range of values, as shown in Figure 2b.

CPGNs are able to include or remove patterns. The results show the success of generated sentences in following the patterns when they are used as constraints. It reveals that inclusive patterns are more informative than exclusive patterns since they provide information about new tokens and their order. In addition, a deeper analysis on the Part-Of-Speech (POS) tag of pattern tokens shows that introducing nouns and adjectives is harder than other tags as they imply more modifications on the neighbouring words (see supplementary materials).

Combination of constraints is challenging for the network. Finally, Table 1 reports that CPGNs are able to combine constraints to achieve better results. Obviously, the more information is provided about the reference, the easier it is to beat the *No-Constraint* model. Nonetheless, depending on the type of constraints, this improvement takes different shapes. Two scenarios can be observed: cooperation or compromise. For instance, when combining the length and BERTScore constraints, $\text{MAE}_{\text{BERTScore}}$ is better than when using the BERTScore constraint alone (from 3.19 to 2.58) while $\text{MAE}_{\text{length}}$ remains close to what is got using the length constraint alone. In this case, the information about the target length guides the system to follow the semantic constraint. The compromise case occurs when each isolated constraint has already achieved very good results on their respective domains. For instance, in the combination of length and inclusive patterns, $\text{MAE}_{\text{length}}$ (respectively $\text{MAE}_{\text{pattern}_{\in}}$) is worse than the one of the length (respectively $\text{pattern}_{\in}$) only model. In parallel, $\text{MAE}_{\text{length}}$ (respectively $\text{MAE}_{\text{pattern}_{\in}}$) is better than the one of the $\text{pattern}_{\in}$ (respectively length) only model. Hence, the system selects a compromise between the two constraints.

6.2 Paraphrase quality

This section comments on Table 1 w.r.t. NLG metrics (BLEU and BERTScore between output and reference) to prove that CPGNs do not blindly apply the constraint, but indeed produce good paraphrases.

Table 2: Examples of CPGNs output given the same source sentence using various types of constraints.

	Source
	so i 'm afraid it was your father who took the grenade from the defense volunteers .
$\mathbf{C_{length}}$	**Output**
5	it was your father .
10	i 'm afraid it was your father 's grenade .
15	i 'm afraid it was your father who took the grenade from defense volunteers .
$\mathbf{C_{top\text{-}2\ parse\ tree}}$	*(source top-2 parse tree: (S(ADVP)(VP)(.)))*
(SBARQ(WHADVP)(SQ))	why did you take the grenade from the defense volunteers ?
(S(PP)(,)(NP)(VP)(.))	by the way , it was your father who took the grenade from the defense volunteers .
(S(S)(,)(CC)(S)(.))	so i 'm afraid it was your father , and he took the grenade from the defense volunteers .
$\mathbf{C_{BERTScore}}$	
0.8	that 's what i 'm afraid of .
0.85	that 's why i 'm afraid of your father .
0.9	that 's why i 'm afraid it 's gon na be your father 's .
0.95	i 'm afraid it was your father who took the grenade from the defense department .
$\mathbf{C_{pattern_{\in}}}$	
* worry * dad *	so i **worry** it was your **dad** who took the grenade from the defense volunteers .
therefore * bomb *	**therefore** , i 'm afraid it was your father who took the grenade from the defense .
* military *	so i 'm afraid it was your father who took the grenade from the **military** .
$\mathbf{C_{pattern_{\notin}}}$	
* afraid *	so i 'm **worried** that it was your father who took the grenade from the defense volunteers .
so * father *	i 'm afraid it was your **dad** 's grenade .
* who * from *	so i 'm afraid it was your father **that** took the grenade **off** the defense .

Structural constraints improve the quality of paraphrases. Considering constraints on length or parse tree show strong improvements across the more lexical based metric BLEU. First, models with length constraint outperform the no-constraint model and the length-aware naive baseline on all metrics. It proves that providing the length constraint is not just a simple template for the network: it learns how to properly use it to generate better paraphrases. Second, parse trees also improve the quality of paraphrases. As expected, using full parse trees generates paraphrases which are very close to the reference as it gives a lot of information. Limiting this information to the top-2 levels of the parse tree still helps but significantly less.

BERTScore constraints enhance semantic quality as well. About semantics, the quality of the generated paraphrase is improved when considering the BERTScore constraint as the $BS(o, i)$ increases in comparison to the No-constraint baseline. However, targeting a given BERTScore is difficult as words are discrete and changing one word inevitably changes the meaning.

Following patterns drastically improves the quality of paraphrases. Patterns provide the model with valuable information about the choice of words and their order. Since CPGNs are successful in following the patterns, improvements are also reported in the quality of paraphrases. For pattern$_{\in}$ constraints, significant improvements are observed on the NLG metrics. They are even sometimes close to the results obtained when using the full parse tree constraint, while much less information is provided by the patterns. Whereas this could be only due to the limited lexical transformations of one or two words in the ParaNMT corpus, the poor results of the naive approaches show that this is not true. Hence, the improvement for pattern$_{\in}$ does not only come from the additional information but also from extra knowledge learned and properly used by the network. Applying pattern$_{\notin}$ results in less improvements especially in terms of BLEU, which is a lexical measure. This is because giving information about which words should be removed does not help to find the same substitution as in the reference.

Combining constraints brings synergy. Among different combinations of constraints, we provided the results of some that we find interesting from an application viewpoint. As shown in the last rows of Table 1, combining two types of constraints is always beneficial in terms of paraphrase quality, w.r.t. using one of each two in isolation. Even if this could be expected since the more information from the reference is provided, the better the results. Especially, the "all types" shows that the model is able to consider up to 5 different constraints with a good final performance.

7 Free constraints results

This section explores the use of free constraints, i.e. when they are *not* derived from the target reference.

Table 3: Constraint satisfaction as MAEs (%, the lower the better) for single or double constraints.

Constraints (c_1 and c_2)	**MAE**$_{c_1}$	**MAE**$_{c_2}$
Length	0.1	-
Top-2 ParseTree	1.2	-
BERTScore	8.3	-
Pattern$_{\in}$	20.9	-
Pattern$_{\notin}$	30.9	-
Length + BERTScore	0.4	8.6
Length + Pattern$_{\in}$	0.1	20.9
BERTScore + Pattern$_{\in}$	11.2	18.6
Length + Top-2 ParseTree	0.8	1.9

Table 4: Results of the human evaluation (✗ = no paraphrase, $\sim$ = ungrammatical paraphrase, ✓ = grammatical paraphrase).

Model	✗	$\sim$	✓
No Constraint	0.64	0.17	0.19
Free Constraint CPGNs			
Length	0.67	0.24	0.09
Top-2 Parse Tree	0.45	0.30	0.25
BERTScore	0.52	0.14	0.34
Pattern$_{\in}$	0.71	0.09	0.20
Pattern$_{\notin}$	0.69	0.10	0.21
Average	0.61	0.17	0.22

7.1 Constraint satisfaction

Table 2 contains some examples of generated paraphrases for various types of constraints, while Table 3 reports constraint satisfaction in terms of MAE for each type of constraint. Note that the TED value for Top-2 Parse Tree is not directly comparable with the one from previous experiments reported in Table 1. Here, TED is computed by comparing the *top-2* levels of the *constraint* and output, whereas, in Table 1, TED compares the *full* trees of the *reference* and output. For other values, scores stay close but a little bit worse. Thereby, CPGNs are still able to follow the constraints even with a larger range of values, including when constraints are combined. Still, two exceptions can be noticed, bringing better results than when using reference-based constraints: Pattern$_{\notin}$ and BERTScore + Pattern$_{\in}$. This will need further investigations.

7.2 Paraphrase quality

As mentioned in Section 5.4, due to the lack of reference for each constraint value in free constraint experiments, NLG metrics cannot be used. For this reason, to measure the quality of free constraint paraphrases, a human evaluation is performed in which workers are asked to rate a paraphrase pair $< x, y >$ on the three-point scale from Kok and Brockett [2010], where x is the source sentence and y is the generated sentence. A value 2 (denoted as ✓) in this scale indicates that y is a grammatical paraphrase of x (i.e., meaning preservation and grammar are OK), while 1 (denoted as $\sim$) means y is an ungrammatical paraphrase of x (only meaning preservation is OK) and 0 (denoted as ✗) means no paraphrase relationship (none of them is OK). Table 4 shows that applying free constraints leads to acceptable paraphrases in terms of quality, mostly in line with scores of the "no-constraint" model. Deeper analysis reveals that syntax-related constraints, i.e., length and parse tree, spoil the grammar more than the others. Second, the model mostly applies the patterns (inclusive or exclusive) with correct grammar. Finally, paraphrasing with the BERTScore constraint is the most challenging of all.

8 Conclusion and future work

In this paper, we have experimented with a general paraphrase generation framework to integrate various types of constraints and thus provide more control on the desired output. The results show that this framework is effectively able to take into account these constraints—although this is with variable performance according to the type of constraints—and that this can help to produce better paraphrases.

Several perspectives are opened by these results. First, the use of patterns seems promising, especially as this matches several applicative needs. Hence, it would be interesting to attempt the use of more complex patterns, e.g. composed of more words or mixing linguistic levels like POS. Likewise, single patterns could be replaced by sets of all the patterns that one would like to see or not see in the output, whenever applicable. Then, the paraphrase models should be shifted to transformer-based architectures, in particular to exploit their ability to easily combine variable numbers of heterogeneous inputs, here the source sentence and the constraints. Finally, it would be interesting to study the integration of CPGNs in a computed-assisted writing tool for end-users, where the remaining challenge would be to help users select or build their constraints.

Acknowledgement

This work has benefited from the financial support of the French National Research Agency (ANR) through the TREMoLo project (ANR-16-CE23-0019)[10].

[10] `http://tremolo.irisa.fr/`

References

Aurélien Max. Local rephrasing suggestions for supporing the work of writers. In Bengt Nordström and Aarne Ranta, editors, *Proceedings of the Advances in Natural Language Processing*, pages 324–335, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-85287-2.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1875–1885, 2018.

Tomoyuki Kajiwara. Negative lexically constrained decoding for paraphrase generation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 6047–6052, 2019.

Cedric Boidin, Verena Rieser, Lonneke van der Plas, Oliver Lemon, and Jonathan Chevelu. Predicting how it sounds: Re-ranking dialogue prompts based on tts quality for adaptive spoken dialogue systems. In *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, 2009.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*, 2019.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. Controlling output length in neural encoder-decoders. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1328–1338, 2016.

Ashutosh Kumar, Kabir Ahuja, Raghuram Vadapalli, and Partha Talukdar. Syntax-guided controlled generation of paraphrases. *Transactions of the Association for Computational Linguistics*, 8:330–345, 2020.

Tanya Goyal and Greg Durrett. Neural syntactic preordering for controlled paraphrase generation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.

Elozino Egonmwan and Yllias Chali. Transformer and seq2seq model for paraphrase generation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 249–255, 2019.

Kees van Deemter, Mariët Theune, and Emiel Krahmer. Real versus template-based natural language generation: A false opposition? *Computational linguistics*, 31(1):15–24, 2005.

Albert Gatt and Ehud Reiter. Simplenlg: A realisation engine for practical applications. In *Proceedings of the European Workshop on Natural Language Generation (ENLG)*, pages 90–93, 2009.

Chris Quirk, Chris Brockett, and William B Dolan. Monolingual machine translation for paraphrase generation. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, pages 142–149, 2004.

Philipp Koehn. *Statistical machine translation*. Cambridge University Press, 2009.

Juncen Li, Robin Jia, He He, and Percy Liang. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1865–1874, 2018.

Shaohan Huang, Yu Wu, Furu Wei, and Zhongzhi Luan. Dictionary-guided editing networks for paraphrase generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6546–6553, 2019.

Di Jin, Zhijing Jin, Zhiting Hu, Olga Vechtomova, and Rada Mihalcea. Deep learning for text style transfer: A survey. *arXiv preprint arXiv:2011.00416*, 2020.

Yunli Wang, Yu Wu, Lili Mou, Zhoujun Li, and Wenhan Chao. Harnessing pre-trained neural networks with rules for formality style transfer. 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 3573–3578, 2019.

Huiyuan Lai, Antonio Toral, and Malvina Nissim. Thank you bart! rewarding pre-trained models improves formality style transfer. *arXiv preprint arXiv:2105.06947*, 2021.

Yu Bao, Hao Zhou, Shujian Huang, Lei Li, Lili Mou, Olga Vechtomova, Xinyu Dai, and Jiajun Chen. Generating sentences from disentangled syntactic and semantic spaces. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 6008–6019, 2019.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. Disentangled representation learning for non-parallel text style transfer. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 424–434, 2019.

Zichao Li, Xin Jiang, Lifeng Shang, and Qun Liu. Decomposable neural paraphrase generation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 3403–3414, 2019.

Ning Dai, Jianze Liang, Xipeng Qiu, and Xuan-Jing Huang. Style transformer: Unpaired text style transfer without disentangled latent representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5997–6007, 2019.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*, 2021.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

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

J Edward Hu, Huda Khayrallah, Ryan Culkin, Patrick Xia, Tongfei Chen, Matt Post, and Benjamin Van Durme. Improved lexically constrained decoding for translation and monolingual rewriting. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 839–850, 2019.

Jonathan Chevelu, Thomas Lavergne, Yves Lepage, and Thierry Moudenc. Introduction of a new paraphrase generation tool based on monte-carlo sampling. In *Proceedings of the Joint Conference of the Annual Meeting of the Association for Computation Linguistics and the International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 249–252, 2009.

Betty Fabre, Tanguy Urvoy, Jonathan Chevelu, and Damien Lolive. Neural-driven search-based paraphrase generation. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 2100–2111, 2021.

Jan Milan Deriu and Mark Cieliebak. Syntactic manipulation for generating more diverse and interesting texts. In *Proceedings of the International Conference on Natural Language Generation (INLG)*, pages 22–34, 2018.

Yue Cao and Xiaojun Wan. Divgan: Towards diverse paraphrase generation via diversified generative adversarial network. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 2411–2421, 2020.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL): system demonstrations*, pages 55–60, 2014.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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 (NAACL-HLT)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

John Wieting and Kevin Gimpel. Paranmt-50m: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 451–462, 2018.

O Bojar, O Dušek, T Kocmi, J Libovickỳ, M Novák, M Popel, R Sudarikov, and D CzEng Variš. Czeng 1.6: Enlarged czech-english parallel corpus with processing tools dockered. In *Proceedings of the International Conference on Text, Speech and Dialogue (TSD)*, 2016.

Sho Takase and Naoaki Okazaki. Positional encoding to control output sequence length. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 3999–4004, 2019.

Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. Controllable paraphrase generation with a syntactic exemplar. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 5972–5984, 2019.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*, 2013.

George A Miller. *WordNet: An electronic lexical database*. MIT press, 1998.

Aaditya Prakash, Sadid A Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. Neural paraphrase generation with stacked residual lstm networks. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 2923–2934, 2016.

Stanley Kok and Chris Brockett. Hitting the right paraphrases in good time. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 145–153, 2010.

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

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

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

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

Vasile Rus and Mihai Lintean. An optimal assessment of natural language student input using word-to-word similarity metrics. In *International Conference on Intelligent Tutoring Systems*, pages 675–676. Springer, 2012.

Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *CoRR*, abs/1706.09799, 2017. URL `http://arxiv.org/abs/1706.09799`.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Sections 6 and 7 thorough comment the positive and negative results.
   (c) Did you discuss any potential negative societal impacts of your work? [No] It is mostly assumed that the constraints would be controlled by humans. Hence, the negative impact would be mainly the cause of the user. Still text generation in general can be an ethical issue in several cases.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A] No theoretical results
   (b) Did you include complete proofs of all theoretical results? [N/A] No theoretical results
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Code is provided in supplementary material and data is a public dataset
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Appendix A.1 details the model architecture.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] The number of experiments prevented from running multiple runs. Additionnally, the size of the dataset (especially, validation and tests) was judged as large enough to provide stable results.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] Nothing special to highlight (standard hardware).

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] As written in Section 3, the model is inspired from Iyyer et al. [2018].

(b) Did you mention the license of the assets? [No] No specific licence was provided with the initial asset. And, the licence for our code is not defined yet (but will be if accepted). ParaNMT-50M uses a non-commercial research purposes licence.

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Code is provided in supplementary material. If accepted, the git repository containing this code will be provided (not given yet due to the anonymity constraint).

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] ParaNMT-50M's licence authorizes free use of the data for non-commercial research purposes.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] We assumed that this has been handled by the authors of ParaNMT-50M.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [No] The human evaluation protocol is described in Section 7.2. Since this protocol is rather simple, no verbatim or screenshot is provided in order to save space. This can be added as an apprendix if judged as important by the reviewers.

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [No] Human evaluators were volunteers.

A Appendix

We provide more details of our constraint-sensitive model in Section A.1. Also, additional evaluation results and analysis are given in Section A.2.

A.1 Detailed model architecture

Given an input sentence $X = [x_1, ..., x_{T_X}]$ and a set of constraints $\mathcal{C} = \{C_i\}_i$ of target characteristics, our goal is to generate a paraphrase sentence $Y = [y_1, ..., y_{T_Y}]$ that satisfies the constraints. Each constraint can be

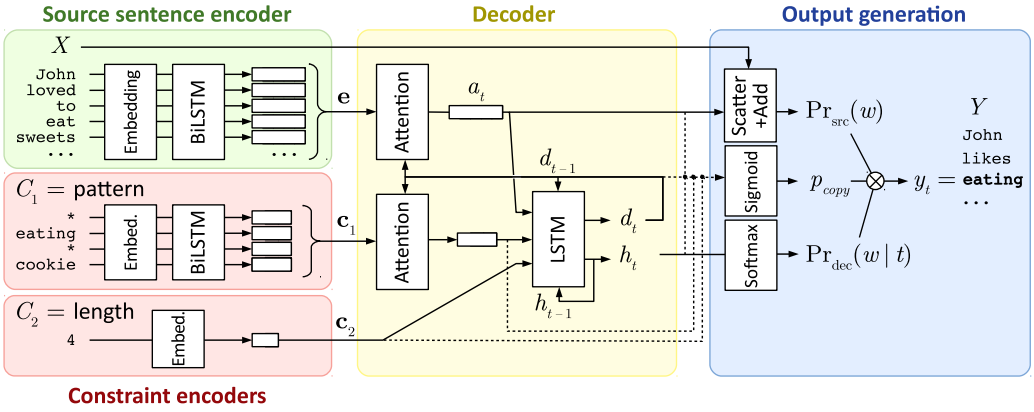


Figure 3: Architecture of the constraint-sensitive model. Some layers have been skipped to optimize readability.

either a single scalar value, i.e., the desired length of output, or can be complex/structured data i.e., a pattern to insert, a syntax tree, etc. In the paper, we refer to this architecture as Constraint-sensitive Paraphrase Generation Network (CPGN).

Inspired by Iyyer et al. [2018], we extend the usual encoder-decoder architecture to include several encoders, one for the source sequence and one for each constraint to apply, as shown in Figure 3. The remainder describes these encoders, the decoder, and the final generation step.

Source sentence encoder Using a single-layer bidirectional LSTM Hochreiter and Schmidhuber [1997], each token x_i of the source sequence X is represented as a word embedding of size 300, and converted into a contextual embedding defined as the encoder hidden state e_i. We denote the full embedding of X as $\mathbf{e} = [e_1, \ldots, e_{T_X}]$.

Constraint encoder In a generic approach, we consider each constraint as a sequence of items. This enables to represent various types of constraints, like word sequences, regular expressions, linearized data structures, scalars, etc. Hence, a constraint C_i is represented as a sequence of N_i elements $[C_{i,j}]_{j \in [\![1, N_i]\!]}$. A dedicated component encodes each element, leading to the constraint embedding $\mathbf{c}_i = [c_{i,j}]_{j \in [\![1, N_i]\!]}$. In practice, several architectures could be used to perform specific encodings but this is not the objective of our work. Here, items are first projected to an embedding space using a fully connected layer, and then fed to a uni-directional LSTM layer. In the case where the constraint is a scalar value (e.g., a desired target length for the output paraphrase), the LSTM layer is skipped as the sequence is of size 1.

Decoder At every time step t, the decoder is conditioned on the source sequence embedding $\mathbf{e}$ and K constraints represented by their embedding $\mathbf{c}_i$. Internal state d_t and hidden state h_t of the decoder yield from the following recurrence:

$$(h_t, d_t) = \textit{decode}(h_{t-1}, d_{t-1}, \mathbf{e}, \mathbf{c_1}, \ldots, \mathbf{c}_K) \; . \tag{1}$$

In practice, the recurrent part is implemented using a uni-directional LSTM layer.

In complement, the input embedding $\mathbf{e}$ is screened through an attention mechanism [Bahdanau et al., 2015] at each time step t, that is:

$$\begin{aligned} \textit{attention}(\mathbf{e})_t = a_t &= \sum_{i=1}^{T_X} \alpha_{t,i} e_i \; , \\ \text{with } \alpha_t &= \textit{softmax}(o_t) \; , \\ \text{and } o_{t,i} &= \textit{tanh}(d_{t-1}, e_i) \; . \end{aligned} \tag{2}$$

Similarly, each constraint embedding c_i is also associated to an attention mechanism inside the decoder. To avoid useless complexity, this mechanism is skipped in the case of scalar constraints as there is only one item on which attention can be paid[11].

Output generation Usually, many parts of a paraphrase directly come from the source sentence, e.g. to handle entity names or out-of-vocabulary words. To enable this behavior, we enrich the model by incorporating the copy mechanism proposed by See et al. [2017]. The principle is that the probability $\Pr(y_t = w)$ to generate an output word w at time step t not only comes from the decoder, but also from the attention paid on the occurrences of w in the source sentence. This is achieved through a linear interpolation as follows:

$$\begin{aligned} \Pr(w|t) = \Pr_{\text{src}}(w) \times p_{copy} \\ + \Pr_{\text{dec}}(w|t) \times (1 - p_{copy}) \; , \end{aligned} \tag{3}$$

$$\text{with } \Pr_{\text{dec}}(w|t) = \textit{softmax}(h_t) \; , \tag{4}$$

$$\Pr_{\text{src}}(w) = \sum_{i \,:\, w_i = w} a_{t,i} \; , \tag{5}$$

$$\text{and} \qquad p_{copy} = \sigma(h_t, d_t, \mathbf{e}, \mathbf{c_1}, \ldots, \mathbf{c}_K) \; . \tag{6}$$

A.2 More evaluation results

Table 5 shows the evaluation results of paraphrase quality in terms of METEOR, ROUGE-L Lin [2004] and GSM Rus and Lintean [2012]. These metrics are computed using the `nlg-eval` package Sharma et al. [2017]. As shown in this table, the trend is similar to the BLEU metric that was reported in the main paper.

In addition, a deeper analysis on the Part-Of-Speech (POS) tag of pattern tokens shows that introducing nouns and adjectives is harder than other tags as they imply more modifications on the neighbouring words (see Figure 4).

[11] Let one note that the generic attention mechanism would not hurt if not skipped since the *softmax* layer would simply result to a singleton weight $\alpha_t = [1]$.

Constraint	Model	BLEU	METEOR	ROUGE-L	GSM	BS(o, r)
None	⋆No-Constraint Baseline	30.0	34.6	61.0	0.84	92.2
Length	Naive Length	23.2	31.8	57.6	0.83	90.5
	LenEmb Kikuchi et al. [2016]	26.2	32.9	59.5	0.82	91.0
	LRPE Takase and Okazaki [2019]	29.4	36.3	62.1	0.86	91.6
	⋆Length	31.6	38.4	62.6	0.85	92.7
Syntax	SCPNIyyer et al. [2018]	25.9	32.7	60.7	0.86	92.1
	⋆Top-2 Parse Tree	30.7	36.6	63.1	0.85	92.7
	⋆Full Parse Tree	52.9	53.1	78.7	0.90	95.5
Semantics	⋆BERTScore	31.0	36.2	62.2	0.85	93.2
Pattern$_{\in}$	Naive Pattern$_{\in}$	22.1	34.3	60.5	0.87	90.4
	Baseline + Naive Pattern$_{\in}$	25.8	36.8	64.5	0.87	91.1
	⋆Pattern$_{\in}$	44.7	52.0	77.9	0.91	94.8
Pattern$_{\notin}$	Naive Pattern$_{\notin}$	19.8	26.5	45.7	0.84	90.1
	Baseline + Naive Pattern$_{\notin}$	21.7	29.3	47.4	0.86	90.8
	⋆Pattern$_{\notin}$	34.7	39.3	66.8	0.87	93.1
Multiple	⋆Length + BERTScore	31.9	38.8	63.2	0.85	93.9
	⋆Length + Top-2 Parse Tree	33.0	57.6	81.4	0.92	93.2
	⋆Length + Pattern$_{\in}$	46.6	57.3	79.2	0.92	95.2
	⋆BERTScore + Pattern$_{\in}$	45.6	53.6	78.8	0.91	95.1
	⋆All types	47.2	55.3	77.0	0.91	94.8

Table 5: Systems performances with various types of constraints from paraphrase quality perspective. The higher the better. BS(o, r) stands for the BERTScore between the output, i.e., the generated paraphrase, and the reference. Models prefixed with ⋆ are based on the proposed CPGN architecture. "All types" includes length, top-2 Parse Tree, BERTScore, pattern$_{\in}$ and pattern$_{\notin}$.

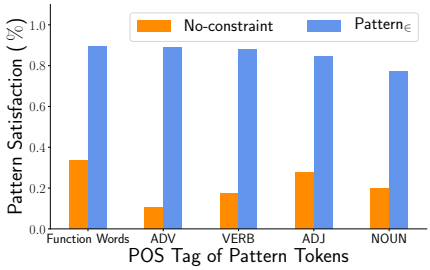


Figure 4: Satisfaction of Pattern$_{\in}$ with respect to POS tags.