Improving Latent Alignment in Text Summarization by Generalizing the Pointer Generator

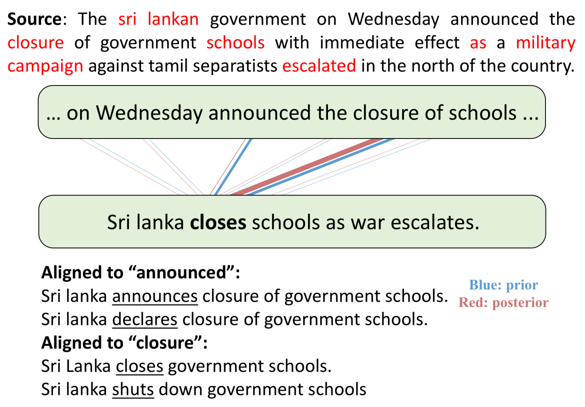
Xiaoyu Shen[1,2]*, **Yang Zhao**[3], **Hui Su**[4] **and Dietrich Klakow**[1]
[1]Spoken Language Systems (LSV), Saarland University, Germany
[2]Max Planck Institute for Informatics, Saarland Informatics Campus, Germany
[3]The University of Tokyo, Japan
[4]Pattern Recognition Center, Wechat AI, Tencent Inc, China

Abstract

Pointer Generators have been the de facto standard for modern summarization systems. However, this architecture faces two major drawbacks: Firstly, the pointer is limited to copying the exact words while ignoring possible inflections or abstractions, which restricts its power of capturing richer latent alignment. Secondly, the copy mechanism results in a strong bias towards extractive generations, where most sentences are produced by simply copying from the source text. In this paper, we address these problems by allowing the model to "edit" pointed tokens instead of always hard copying them. The editing is performed by transforming the pointed word vector into a target space with a learned relation embedding. On three large-scale summarization dataset, we show the model is able to (1) capture more latent alignment relations than exact word matches, (2) improve word alignment accuracy, allowing for better model interpretation and controlling, (3) generate higher-quality summaries validated by both qualitative and quantitative evaluations and (4) bring more abstraction to the generated summaries.[1]

1 Introduction

Modern state-of-the-art (SOTA) summarization models are built upon the pointer generator architecture (See et al., 2017). At each decoding step, the model generates a sentinel to decide whether to sample words based on the neural attention (generation mode), or directly copy from an aligned source context (point mode) (Gu et al., 2016; Merity et al., 2017; Yang et al., 2017). Though outperforming the vanilla attention models, the pointer generator only captures exact word matches. As shown in Fig. 1, for abstractive summarization,



Source: The sri lankan government on Wednesday announced the closure of government schools with immediate effect as a military campaign against tamil separatists escalated in the north of the country.

... on Wednesday announced the closure of schools ...
Sri lanka **closes** schools as war escalates.
Blue: prior
Red: posterior

Aligned to "announced":
Sri lanka announces closure of government schools.
Sri lanka declares closure of government schools.
Aligned to "closure":
Sri Lanka closes government schools.
Sri lanka shuts down government schools

Figure 1: Alignment visualization of our model when decoding "closes". Posterior alignment is more accurate for model interpretation. In contrast, the prior alignment probability is spared to "announced" and "closure", which can be manually controlled to generate desired summaries. Decoded samples are shown when aligned to "announced" and "closure" respectively. Highlighted source words are those that can be directly aligned to a target token in the gold summary.

there exists a large number of syntactic inflections (escalated → escalates) or semantic transformations (military campaign → war), where the target word also has an explicit grounding in the source context but changes its surface. In standard pointer generators, these words are not covered by the point mode. This largely restricts the application of the pointer generator, especially on highly abstractive dataset where only a few words are exactly copied. Moreover, the hard copy operation biases the model towards extractive summarization, which is undesirable for generating more human-like summaries (Kryściński et al., 2018).

To solve this problem, we propose **G**eneralized **P**ointer **G**enerator (**GPG**) which replaces the hard copy component with a more general soft "editing" function. We do this by learning a relation embedding to transform the pointed word into a

*Correspondence to xshen@mpi-inf.mpg.de
[1]The source code is available at https://github.com/chin-gyou/generalized-PG.

target embedding. For example, when decoding "closes" in Figure 1, the model should first point to "closure" in the source, predict a relation to be applied (noun $\rightarrow$ third person singular verb), then transform "closure" into "closes" by applying the relation transformation. The generalized point mode is encouraged to capture such latent alignment which cannot be identified by the standard pointer generator.

This improved alignment modelling is intriguing in that (a) people can better control the generation by manipulating the alignment trajectory, (b) posterior alignment can be inferred by Bayes' theorem (Deng et al., 2018; Shankar and Sarawagi, 2019) to provide a better tool for interpretation[2] and finally (c) explicitly capturing the alignment relation should improve generation performance. (Figure 1 shows an example of how latent alignment can improve the controllability and interpretation. Pointer generators fail to model such alignment relations that are not exact copies.) To eliminate the OOV problem, we utilize the byte-pair-encoding (BPE) segmentation (Sennrich et al., 2016) to split rare words into sub-units, which has very few applications in summarization so far (Fan et al., 2018; Kiyono et al., 2018), though being a common technique in machine translation (Wu et al., 2016; Vaswani et al., 2017; Gehring et al., 2017).

Our experiments are conducted on three summarization datasets: CNN/dailymail (Hermann et al., 2015), English Gigaword (Rush et al., 2015) and XSum (Narayan et al., 2018) (a newly collected corpus for extreme summarization). We further perform human evaluation and examine the word alignment accuracy on the manually annotated DUC 2004 dataset. Overall we find our model provides the following benefits:

1. It can capture richer latent alignment and improve the word alignment accuracy, enabling better controllability and interpretation.

2. The generated summaries are more faithful to the source context because of the explicit alignment grounding.

3. It improves the abstraction of generations because our model allows editing the pointed token instead of always copying exactly.

In the next section, we will first go over the background, then introduce our model and finally present the experiment results and conclusion.

2 Background

Let X, Y denote a source-target pair where X corresponds to a sequence of words $x_1, x_2, \ldots, x_n$ and Y is its corresponding summary $y_1, y_2, \ldots, y_m$. In this section, we introduce two baseline models for automatically generating Y from X:

2.1 Seq2seq with Attention

In a seq2seq model, each source token x_i is encoded into a vector h_i. At each decoding step t, the decoder computes an attention distribution a_t over the encoded vectors based on the current hidden state d_t (Bahdanau et al., 2015):

$$a_t = \text{softmax}(f(h_i, d_t)) \tag{1}$$

f is a score function to measure the similarity between h_i and d_t. The context vector c_t and the probability of next token are computed as below.

$$\begin{aligned} c_t &= \sum_i h_i a_{t,i} \\ y_t^* &= [d_t \circ c_t] L \\ p_{vocab} &= \text{softmax}(y_t^* W^T) \end{aligned} \tag{2}$$

$\circ$ means concatenation and L, W are trainable parameters. We tie the parameters of W and the word embedding matrix as in Press and Wolf (2017); Inan et al. (2017). Namely, a target vector y_t^* is predicted, words having a higher inner product with y_t^* will have a higher probability.

2.2 Pointer Generator

The pointer generator extends the seq2seq model to support copying source words (Vinyals et al., 2015). At each time step t, the model first computes a generation probability $p_{gen} \in [0, 1]$ by:

$$p_{gen} = \sigma(\text{MLP}_g([d_t \circ c_t])) \tag{3}$$

σ is a sigmoid function and MLP_g is a learnable multi-layer perceptron. p_{gen} is the probability of enabling the generation mode instead of the point

[2]The induced alignment offers useful annotations for people to identify the source correspondence for each target word. News editors can post-edit machine-generated summaries more efficiently with such annotation. For summary readers, it also helps them track back the source context when they are interested in some specific details. Derivation for inducing the posterior alignment is in the appendix A.1.

mode. In the generation mode, the model computes the probability over the whole vocabulary as in Eq. 2. In the point mode, the model computes which source word to copy based on the attention distribution a_t from Eq.1. The final probability is marginalized over $a_{t,i}$:

$$p(y_t) = p_{gen}p_{vocab}(y_t) + (1 - p_{gen}) \sum_i a_{t,i}\delta(y_t|x_i)$$

$$\delta(y_t|x_i) = \begin{cases} 1, & \text{if } y_t = x_i. \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

If we know exactly from which mode each word comes from, *e.g.*, by assuming all co-occurred words are copied, then the marginalization can be omitted (Gulcehre et al., 2016; Wiseman et al., 2017), but normally p_{gen} is treated as a latent variable (Gu et al., 2016; See et al., 2017).

3 Generalized Pointer Generator (GPG)

As seen in Eq .4, $\delta(y_t|x_i)$ is a 0-1 event that is only turned on when y_t is exactly the same word as x_i. This restricts the expressiveness of the point mode, preventing it from paying attention to inflections, POS transitions or paraphrases. This section explains how we generalize pointer networks to cover these conditions.

Redefine $\boldsymbol{\delta(y_t|x_i)}$: We extend $\delta(y_t|x_i)$ by defining it as a smooth probability distribution over the whole vocabulary. It allows the pointer to edit x_i to a different word y_t instead of simply copying it. Following Eq. 2, we derive $\delta(y_t|x_i)$ by first predicting a target embedding $y^*_{t,i}$, then applying the softmax. The difference is that we derive $y^*_{t,i}$ as the summation of the pointed word embedding $\overrightarrow{x_i}$ and a relation embedding $r(d_t, h_t)$:

$$\begin{aligned} y^*_{t,i} &= r(d_t, h_i) + \overrightarrow{x_i} \\ \delta(y_t|x_i) &= \text{softmax}(y^*_{t,i}W^T) \end{aligned} \tag{5}$$

$\overrightarrow{x_i}$ denotes the embedding of the word x_i. $r(d_t, h_t)$ can be any function conditioning on d_t and h_t, which we parameterize with a multi-layer-perceptron in our experiments. The computation of $y^*_{t,i}$ is similar to the classical TransE model (Bordes et al., 2013) where an entity vector is added by a relation embedding to translate into the target entity. The intuition is straightforward: After pointing to x_t, humans usually first decide which relation should be applied (inflection, hypernym, synonym, etc) based on the context $[d_t, h_i]$, then transform x_i to the proper target word y_t. Using addition transformation is backed by the observation that vector differences often reflect meaningful word analogies (Mikolov et al., 2013; Pennington et al., 2014) ("*man*" − "*king*" ≈ "*woman*" − "*queen*") and they are effective at encoding a great amount of word relations like hypernym, meronym and morphological changes (Vylomova et al., 2016; Hakami et al., 2018; Allen and Hospedales, 2019). These word relations reflect most alignment conditions in text summarization. For example, humans often change the source word to its hypernym (boy $\rightarrow$ child), to make it more specific (person $\rightarrow$ man) or apply morphological transformations (liked $\rightarrow$ like). Therefore, we assume $\delta(y_t|x_i)$ can be well modelled by first predicting a relation embedding to be applied, then added to $\overrightarrow{x_i}$. If x_i should be exactly copied like in standard pointer generators, the relation embedding is a zero vector meaning an identity transition. We also tried applying more complex transitions to $\overrightarrow{x_i}$ like diagonal mapping (Trouillon et al., 2016), but did not observe improvements. Another option is to estimate $\delta(y_t|x_i)$ directly from (d_t, h_i) by an MLP regardless of $\overrightarrow{x_i}$. However, this leads to poor alignment and performance drop because y_t is not explicitly grounded on x_i[3]. A comparison of different choices can be found in the appendix A.4. In this paper, we stick to Eq. 5 to compute $\delta(y_t|x_i)$.

Estimate Marginal Likelihood: Putting Eq. 5 back to Eq. 4, the exact marginal likelihood is too expensive for training. The complexity grows linearly with the source text length n and each computation of $\delta(y_t|x_i)$ requires a separate softmax operation. One option is to approximate it by sampling like in hard attention models (Xu et al., 2015; Deng et al., 2017), but the training becomes challenging due to the non-differentiable sampling process. In our work, we take an alternative strategy of marginalizing only over k most likely aligned source words. This top-k approximation is widely adopted when the target distribution is expected to be sparse and only a few modes dominate (Britz et al., 2017; Ke et al., 2018; Shankar et al., 2018). We believe this is a valid assumption in text summarization since most source

[3] h_i can contain context information from surrounding words and thus not necessarily relates to word x_i. It is part of the reason that neural attention has a poor alignment (Koehn and Knowles, 2017). Making it grounded on x_i improves alignment and performance is also observed in machine translation (Nguyen and Chiang, 2018; Kuang et al., 2018)

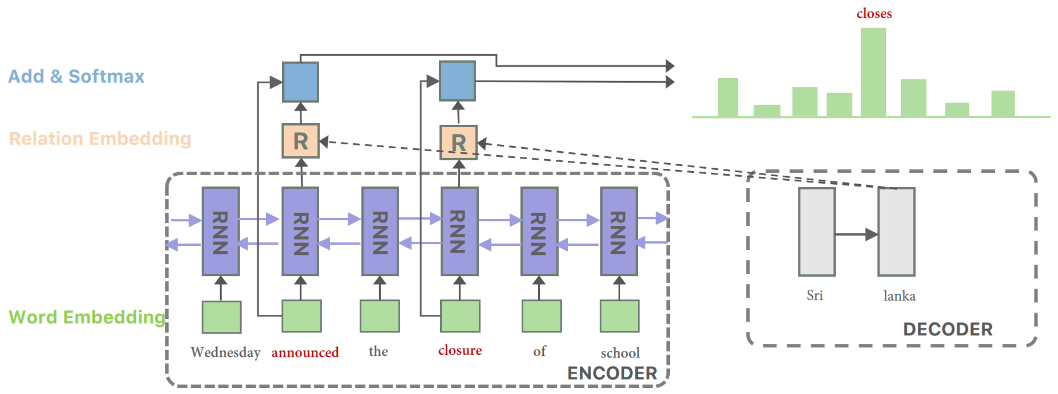


Figure 2: Architecture of the generalized pointer. The same encoder is applied to encode the source and target. When decoding "closes", we first find top-k source positions with the most similar encoded state. For each position, the decoding probability is computed by adding its word embedding and a predicted relation embedding.

tokens have a vanishingly small probability to be transferred into a target word.

For each target word, how to determine the k most likely aligned source words is crucial. An ideal system should always include the gold aligned source word in the top-k selections. We tried several methods and find the best performance is achieved when encoding each source/target token into a vector, then choosing the k source words that are closest to the target word in the encoded vector space. The closeness is measured by the vector inner product[4]. The encoded vector space serves like a contextualized word embedding (McCann et al., 2017; Peters et al., 2018). Intuitively if a target word can be aligned to a source word, they should have similar semantic meaning and surrounding context thus should have similar contextualized word embeddings. The new objective is then defined as in Eq. 6:

$$\begin{aligned} p(y_t) &= p_{gen} p_{vocab}(y_t) + (1 - p_{gen}) p_{point}(y_t) \\ p_{point}(y_t) &= \sum_i a_{t,i} \delta(y_t | x_i) \\ &\approx \sum_{i; h_i^T e(y_t) \in \text{TopK}} a_{t,i} \delta(y_t | x_i) \end{aligned} \tag{6}$$

$e(y_t)$ is the encoded vector for y_t. The marginalization is performed only over the k chosen source words. Eq. 6 is a lower bound of the data likelihood because it only marginalizes over a subset of X. In general a larger k can tighten the bound to get a more accurate estimation and we analyze the effect of k in Section 5.2. Note that the only extra parameters introduced by our model are the multi-layer-perceptron to compute the relation embedding $r(d_t, h_i)$. The marginalization in Eq. 6 can also be efficiently parallelized. An illustration of the generalized pointer is in Figure 2.

[4]We compared several strategies for choosing the top-k words and report it in Appendix A.5. Note that the top-k approximation is only used for training, so we can spot the whole target text to decide top-k candidates.

4 Related Work

Neural attention models (Bahdanau et al., 2015) with the seq2seq architecture (Sutskever et al., 2014) have achieved impressive results in text summarization tasks. However, the attention vector comes from a weighted sum of source information and does not model the source-target alignment in a probabilistic sense. This makes it difficult to interpret or control model generations through the attention mechanism. In practice, people do find the attention vector is often blurred and suffers from poor alignment (Koehn and Knowles, 2017; Kiyono et al., 2018; Jain and Wallace, 2019). Hard alignment models, on the other hand, explicitly models the alignment relation between each source-target pair. Though theoretically sound, hard alignment models are hard to train. Exact marginalization is only feasible for data with limited length (Yu et al., 2016; Aharoni and Goldberg, 2017; Deng et al., 2018; Backes et al., 2018), or by assuming a simple copy generation process (Vinyals et al., 2015; Gu et al., 2016; See et al., 2017). Our model can be viewed as a combination of soft attention and hard alignment, where a simple top-k approximation is used to train the alignment part (Shankar et al., 2018; Shankar and Sarawagi, 2019). The hard alignment generation probability is designed as a relation summation operation to better fit the summarization task. In this way, the generalized copy mode acts as a hard alignment component to capture the direct word-to-word transitions. On the contrary, the generation mode is a standard soft-attention structure to only model words that are purely functional, or need fusion, high-level inference and can be hardly aligned to any specific source context (Daumé III and Marcu, 2005).

5 Experiments and Results

In the experiment, we compare seq2seq with attention, standard pointer generators and the proposed generalized pointer generator (GPG). To further analyze the effect of the generalized pointer, we implement a GPG model with only the point mode (GPG-ptr) for comparison. We first introduce the general setup, then report the evaluation results and analysis.

5.1 General Setup

Dataset: We perform experiments on the CNN/dailymail (Hermann et al., 2015), English Gigaword (Rush et al., 2015) and XSum (Narayan et al., 2018) dataset. We put statistics of the datasets in Appendix A.2. CNN/DM contains online news with multi-sentence summaries (We use the non-anonymized version from See et al. (2017)). English Gigaword paired the first sentence of news articles with its headline. XSum corpus provides a single-sentence summary for each BBC long story. We pick these three dataset as they have different properties for us to compare models. CNN/DM strongly favors extractive summarization (Kryściński et al., 2018). Gigaword has more one-to-one word direct mapping (with simple paraphrasing) (Napoles et al., 2012) while XSum needs to perform more information fusion and inference since the source is much longer than the target (Narayan et al., 2018).

Model: We use single-layer bi-LSTM encoders for all models. For comparison, hidden layer dimensions are set the same as in Zhou et al. (2017) for Gigaword and See et al. (2017) for CNN/DM and XSum. We train with batch size 256 for gigaword and 32 for the other two. The vocabulary size is set to 30k for all dataset. Word representations are shared between the encoder and decoder. We tokenize words with WordPiece segmentation (Wu et al., 2016) to eliminate the OOV problem. More details are in Appendix A.3

Inference: We decode text using beam search (Graves, 2012) with beam size 10. We apply length normalization to rescale the score. Unlike See et al. (2017); Gehrmann et al. (2018), we do not explicitly impose coverage penalty since it brings extra hyper-parameters. Instead, for CNN/Dailymail, we use a simple tri-gram penalty (Paulus et al., 2018) to prevent repeated generations. GPG models use an exact marginalization for testing and decoding, while for training and validation we use the top-k approximation mentioned above. The decoder will first decode sub-word ids then map them back to the normal sentence. All scores are reported on the word level and thus comparable with previous results. When computing scores for multi-sentence summaries. The generations are split into sentences with the NLTK sentence tokenizer.

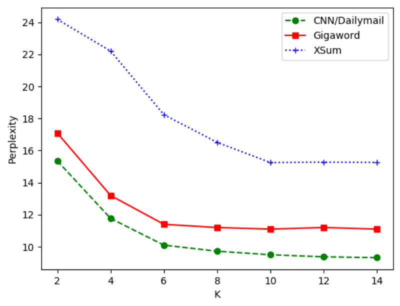


Figure 3: Test perplexity when increasing k

5.2 Results and Analysis

The results are presented in the following order: We first study the effect of the hyperparameter k, then evaluate model generations by automatic metrics and look into the generation's level of abstraction. Finally, we report the human evaluation and word alignment accuracy.

Effect of K: k is the only hyperparameter introduced by our model. Figure 3 visualizes the effect of k on the test perplexity. As mentioned in Sec 3, a larger k is expected to tighten the estimation bound and improve the performance. The figure shows the perplexity generally decreases as increasing k. The effect on Gigaword and XSum saturates at $k = 6$ and 10 respectively, so we fix such k value for later experiments. For the longer dataset CNN/Dailymail, the perplexity might still decrease a bit afterwards, but the improvement is marginal, so we set $k = 14$ for the memory limit.

Automatic Evaluation: The accuracy is evaluated based on the standard metric ROUGE (Lin,

Method	R-1	R-2	R-L	PPL
Point.Gen.+Cov*	39.53	17.28	36.38	
Bottom Up[†]	**41.22**	**18.68**	**38.34**	
seq2seq	39.79	17.37	36.34	17.49
Point.Gen.	40.03	17.52	36.77	12.36
GPG-ptr	40.54	**18.05**	37.19	10.23
GPG	**40.95**	18.01	**37.46**	**9.37**

Table 1: ROUGE score on CNN/Dailymail. * marks results from See et al. (2017), and † from Gehrmann et al. (2018). Underlined values are significantly better than Point.Gen. with $p = 0.05$.

Method	R-1	R-2	R-L	PPL
seq2seq*	34.04	15.95	31.68	
DRGD[†]	**36.27**	**17.57**	**33.62**	
seq2seq	36.01	17.52	33.60	18.92
Point.Gen.	36.14	17.68	33.56	14.90
GPG-ptr	37.14	**19.05**	**34.67**	12.32
GPG	**37.23**	19.02	34.66	**11.41**

Table 2: ROUGE score on Gigaword. * marks results from the word-based seq2seq implementation of Zhou et al. (2017), and † from Li et al. (2017).

Method	R-1	R-2	R-L	PPL
Point.Gen*	29.70	9.21	23.24	
T-CONVS2S*	**31.89**	**11.54**	**25.75**	
seq2seq	31.90	11.15	25.48	22.87
Point.Gen.	31.87	11.20	25.42	17.83
GPG-ptr	31.49	11.02	25.37	18.62
GPG	**33.11**	**12.55**	**26.57**	**15.28**

Table 3: ROUGE score on XSum. * marks results from Narayan et al. (2018). Underlined values are significantly better than Point.Gen. with $p = 0.05$.

2004) and the word perplexity on the test data. We report the ROUGE-1, ROUGE-2 and ROUGE-L F-score measured by the official script. Table 1, 2 and 3 lists the results for CNN/Dailymail, Gigaword and XSum respectively. Statistically significant results are underlined[5]. On the top two rows of each table, we include two results taken from current state-of-the-art word-based models. They are incomparable with our model because of the different vocabulary, training and decoding process, but we report them for completeness. Lower rows are results from our implemented models. Pointer generators bring only slight improvements over the seq2seq baseline. This suggests that after eliminating the OOV problem, the naive seq2seq with attention model can already implicitly learn most copy operations by itself. GPG models outperform seq2seq and pointer generators on all dataset. The improvement is more significant for more abstractive corpus Gigaword and XSum, indicating our model is effective at identifying more latent alignment relations.

Notably, even the pure pointer model (GPG-ptr) without the generation mode outperforms standard pointer generators in CNN/DM and Gigaword, implying most target tokens can be generated by aligning to a specific source word. The finding is consistent with previous research claiming CNN/DM summaries are largely extractive (Zhang et al., 2018; Kryściński et al., 2018). Though the Gigaword headline dataset is more abstractive, most words are simple paraphrases of some specific source word, so pure pointer GPG-ptr can work well. This is different from the XSum story summarization dataset where many target words require high-level abstraction or inference and cannot be aligned to a single source word, so combining the point and generation mode is necessary for a good performance.

The word perplexity results are consistent over all dataset (GPG $<$ GPG-ptr $<$ Point.Gen. $<$ seq2seq). The reduction of perplexity does not necessarily indicate an increase for the ROUGE score, especially for pointer generators. This might attribute to the different probability computation of pointer generators, where the probability of copied words are only normalized over the source words. This brings it an inherent advantage over other models where the normalization is over the whole 30k vocabularies.

Level of Abstraction: In Tab. 4, we look into how abstractive the generated summaries are by calculating the proportion of novel unigram, bigram and trigrams that do not exist in the corresponding source text. On CNN/DM, as the generations contain multiple sentences, we further report the proportion of novel sentences (obtained with NLTK sent_tokenize).

Tab. 4 reflects the clear difference in the level of abstraction (seq2seq $>$ GPG $>$ GPG-ptr $>$ Point.Gen.). Though the seq2seq baseline generated most novel words, many of them are hallu-

[5]Results on the Gigaword test set is not significant due to the smalle test size (1951 article-summary pairs).

Models	% of NNs in CNN/Dailymail				% of NNs in Gigaword			% of NNs in XSum		
	NN-1	**NN-2**	**NN-3**	**NN-S**	**NN-1**	**NN-2**	**NN-3**	**NN-1**	**NN-2**	**NN-3**
Seq2seq	0.38	3.56	7.98	54.97	16.15	52.84	73.76	27.05	76.54	92.07
Point.Gen.	0.04	1.51	4.29	35.82	13.99	47.79	68.53	19.45	66.68	84.59
GPG-ptr	0.17	2.05	5.08	41.64	14.05	48.09	70.70	20.03	69.54	87.14
GPG	0.35	2.91	5.66	49.24	15.14	52.07	72.73	24.16	71.93	87.94
Reference	9.08	46.71	67.99	97.78	48.26	84.53	94.43	32.24	84.12	95.92

Table 4: Proportion of novel n-grams (NN-1,2,3) and sentences (NN-S) on generated summaries. GPG generate more novel words compared with standard pointer generators, though still slightly lower than seq2seq.

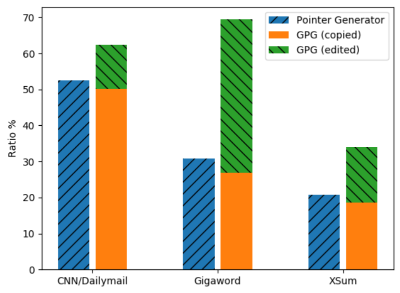


Figure 4: Pointing Ratio of the standard pointer generator and GPG (evaluated on the test data). GPG enables the point mode more often, but quite a few pointed tokens are edited rather than simply copied.

Article: (...) and bild **reported** that the (...)
Summary: (...) and bild **report** that the (...) [0.964]

Article: (...) thousands more **death** row prisoner (...)
Summary: (...) thousands more **deaths** row (...) [0.981]

Article: (...) surviving **relatives** of a woman who (...)
Summary: (...) **family** of woman who (...) [0.984]

Article: **jordan** 's crown prince (...)
Summary: **jordanian** crown prince (...) [0.993]

Article: a middle-aged man and a young **girl** (...) [0.814]
Summary: a man and a **child** died (...)

Article: (...) , was **abducted** in 2012 (...)
Summary: (...) was **kidnapped** in (...) [0.924]

Figure 5: Examples of summaries produced by GPG. Each two samples from CNN/DM, Gigaword and XSum (up to down). **bold** denotes novel words and their pointed source tokens. Bracketed numbers are the pointing probability $(1 - p_{gen})$ during decoding.

cinated facts (see Fig 6), as has also been noted in See et al. (2017). The abstraction of GPG model is close to seq2seq and much higher than copy-based pointer generators. We believe it comes from their ability of "editing" pointed tokens rather than simple copying them.

To examine the pointing behavior of the GPG model, we visualize the average pointing ratio on three dataset in Fig. 4. The pointing ratio can be considered as the chance that a word is generated from the point mode instead of the generation mode. We compute it as $(1 - p_{gen}) \sum_i a_{t,i} \delta(y_t|x_i)/p(y_t)$, averaged over all target tokens in the test data. For the GPG model, we further split it into the copy ratio (words that are exactly copied) and the editing ratio (words that are edited). We find the GPG model enables the point mode more frequently than standord pointer generators, especially on the Gigaword dataset (40% more). This also explains why a pure pointer model is more effective for Gigaword and CNN/DM. More than 60% target tokens can be generated from the point mode, while for XSum the ratio is less than 40%. Quite a few pointing operation includes text rewriting (green bar in Fig. 4). This could explain why our model is able to generate more novel words.

A few examples are displayed in Fig 5. We find our model frequently changes the tense (reported $\rightarrow$ report), singular/plural (death $\rightarrow$ deaths) or POS tag (jordan $\rightarrow$ jordanian) of words. Sometimes it also paraphrases (relatives $\rightarrow$ family) or abstracts a noun to its hypernym (girl $\rightarrow$ child). The word editing might be wrong though. For example, "death row prisoner" is wrongly changed to "deaths row prisoner" in the second example, possibly because this phrase is rarely seen so that the model made an error by mistaking "death" as the main subject after "thousands more"[6].

Human evaluation: We further perform a human evaluation to assess the model generations. We focus on evaluating the fluency and faithfulness since the ROUGE score often fails to quan-

[6]It also reveals a limit of GPG model in that it only models token-level alignment. For phrases like death row prisoner, it cannot align it based on its compositional meaning.

	Fluency	Faithfulness	0/1
seq2seq	**0.83**	0.61	0.53
Point.Gen.	0.78	0.65	0.55
GPG-ptr	0.79	**0.78**	0.67
GPG	0.82	0.74	**0.69**
Gold	0.96	0.92	0.96

Table 5: Human evaluation results on DUC 2004. 0/1 is the score for the 0/1 Turing test.

tify them (Schluter, 2017; Cao et al., 2018). 100 random source-target pairs are sampled from the human-written DUC 2004 data for task 1&2 (Over et al., 2007). Models trained on Gigaword are applied to generate corresponding summaries. The gold targets, together with the model generations are randomly shuffled then assigned to 10 human annotators. Each pair is evaluated by three different people and the most agreed score is adopted. Each pair is assigned a 0-1 score to indicate (1) whether the target is fluent in grammar, (2) whether the target faithfully conveys the source information without hallucination and (3) whether the target is considered human-generated or machine-generated (like a 0/1 Turing test). The averaged score is reported in Table 5. All models generally achieve high scores in fluency, but generations from GPG models are more faithful to the source information thereby have a larger chance of fooling people into believe they're human-generated (over 0.1 higher score on the 0/1 Turing test). This can be explained by GPG's capability at capturing more latent alignments. As shown in Figure 4, GPG generates over half of the target words by its point mode. Words are generated by explicitly grounding on some source context instead of fabricating freely.

Fig. 6 compares some generation snippets. As can be observed, seq2seq models tend to freely synthesize wrong facts not grounded on the source text, especially on the more difficult XSum dataset. In the last example, seq2seq only capture the subject "odom" and some keywords "police", "basketball" then start to freely fabricate random facts. Pointer generators are slightly better as it is trained to directly copy keyword from the source. However, once it starts to enter the generation mode ("of british" in example 2 and "has been arrested" in example 3), the generation also loses control. GPG largely alleviates the problems because it can point to an aligned source word, then transform it by a learned relation embedding. The explicit alignment modelling encourages the model to stay close to the source information.

	seq2seq	Point.Gen.	GPG
Prec	0.361	0.435 (0.512)	0.533 (0.628)

Table 6: Word Alignment Precision on DUC 2004. Number in bracket is the posterior alignment precision.

Alignment Accuracy: We also manually annotate the word alignment on the same 100 DUC 2004 pairs. Following Daumé III and Marcu (2005), words are allowed to be aligned with a specific source word, phrase or a "null" anchor meaning that it cannot be aligned with any source word. The accuracy is only evaluated on the target words with a non-null alignment. For each target token, the most attended source word is considered as alignment (Ghader and Monz, 2017). For the pointer generator and GPG, we also induce the posterior alignment by applying the Bayes' theorem (derivation in appendix A.1). We report the alignment precision (Och and Ney, 2000) in Table 6, i.e., an alignment is considered as valid if it matches one of the human annotated ground truth.

The results show that GPG improves the alignment precision by 0.1 compared with the standard pointer generator. The posterior alignment is more accurate than the prior one (also reflected in Figure 1), enabling better human interpretation.

6 Conclusion

In this work, we propose generalizing the pointer generator to go beyond exact copy operation. At each decoding step, the decoder can either generate from the vocabulary, copy or edit some source words by estimating a relation embedding. Experiments on abstractive summarization show the generalized model generates more abstract summaries yet faithful to the source information. The generalized pointer is able to capture richer latent alignment relationship beyond exact copies. This helps improve the alignment accuracy, allowing better model controllability and interpretation.

We believe the generalized pointer mechanism could have potential applications in many fields where tokens are not exactly copied. By integrating off-the-shelf knowledge bases to clearly model the transition relation embedding, it should further improve the interpretability and might be espe-

Article: (...) marseille prosecutor brice robin told cnn that " so far no videos were used in the crash investigation . " he added , " a person who has such a video needs to immediately give it to the investigators . " robin 's comments follow claims by two magazines , german daily bild and french paris match (...)
Seq2seq: marseille prosecutor brice robin tells cnn that " so far no videos were used in the crash investigation "
Point.Gen: robin 's comments follow claims by two magazines , german daily bild and french (..)
GPG: " so far no videos were used in the crash investigation , " prosecutor brice robin says (..)

Article: surviving relatives of a woman who claimed she was raped ## years ago by the british queen 's representative in australia are seeking to withdraw a lawsuit against him , after the case drew widespread publicity in australia .
Seq2seq: family of british queen 's representative in australia seeking to withdraw lawsuit against him .
Point.Gen: surviving relatives of british queen 's representative seeking to withdraw lawsuit against him .
GPG: family of woman who claimed she was victim of british queen 's representative seeks to withdraw lawsuit .

Article: police were called to love ranch brothel in crystal , nevada , after he was found unresponsive on tuesday . the american had to be driven to hospital (...) mr odom , 35 , has played basketball for (...) lakers and clippers . he (...) was suspended from the nba for violating its anti-drug policy (...) was named nba sixth man of the year (...)
Seq2seq: basketball legend odom odom has died at the age of 83 , police have confirmed .
Point.Gen: a former nba sixth man has been arrested on suspicion of anti-drug offences in the us state of california .
GPG: the american basketball association (lakers) star lamar odom has been found unconscious in the us state of nevada .

Figure 6: Examples of generated summaries. Examples are taken from CNN/DM, Gigaword and XSum (from up to down). Darker means higher pointing probability.

cially helpful under low-resource settings, which we leave for future work.

Acknowledgments

We thank anonymous reviewers for valuable comments, thank Marius Mosbach for the proof reading. Xiaoyu Shen is supported by IMPRS-CS fellowship. The work is partially funded by DFG collaborative research center SFB 1102

References

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2004–2015.

Carl Allen and Timothy Hospedales. 2019. Analogies explained: Towards understanding word embeddings. *ICML*.

Michael Backes, Pascal Berrang, Mathias Humbert, Xiaoyu Shen, and Verena Wolf. 2018. Simulating the large-scale erosion of genomic privacy over time. *IEEE/ACM transactions on computational biology and bioinformatics*, 15(5):1405–1412.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Denny Britz, Melody Guan, and Minh-Thang Luong. 2017. Efficient attention using a fixed-size memory representation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 392–400.

Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2018. Faithful to the original: Fact aware neural abstractive summarization. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98.

Hal Daumé III and Daniel Marcu. 2005. Induction of word and phrase alignments for automatic document summarization. *Computational Linguistics*, 31(4):505–530.

Yuntian Deng, Anssi Kanervisto, Jeffrey Ling, and Alexander M Rush. 2017. Image-to-markup generation with coarse-to-fine attention. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 980–989. JMLR. org.

Yuntian Deng, Yoon Kim, Justin Chiu, Demi Guo, and Alexander Rush. 2018. Latent alignment and variational attention. In *Advances in Neural Information Processing Systems*, pages 9735–9747.

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

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*, pages 1243–1252.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109.

Hamidreza Ghader and Christof Monz. 2017. What does attention in neural machine translation pay attention to? In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 30–39.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.

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

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

Huda Hakami, Kohei Hayashi, and Danushka Bollegala. 2018. Why does pairdiff work?-a mathematical analysis of bilinear relational compositional operators for analogy detection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2493–2504.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. *ICLR*.

Sarthak Jain and Byron C Wallace. 2019. Attention is not explanation. *NAACL*.

Nan Rosemary Ke, Anirudh Goyal ALIAS PARTH GOYAL, Olexa Bilaniuk, Jonathan Binas, Michael C Mozer, Chris Pal, and Yoshua Bengio. 2018. Sparse attentive backtracking: Temporal credit assignment through reminding. In *Advances in Neural Information Processing Systems*, pages 7640–7651.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *ICLR*.

Shun Kiyono, Sho Takase, Jun Suzuki, Naoaki Okazaki, Kentaro Inui, and Masaaki Nagata. 2018. Unsupervised token-wise alignment to improve interpretation of encoder-decoder models. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 74–81.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39.

Wojciech Kryściński, Romain Paulus, Caiming Xiong, and Richard Socher. 2018. Improving abstraction in text summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1808–1817.

Shaohui Kuang, Junhui Li, António Branco, Weihua Luo, and Deyi Xiong. 2018. Attention focusing for neural machine translation by bridging source and target embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1767–1776.

Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. 2017. Deep recurrent generative decoder for abstractive text summarization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2091–2100.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

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

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. *ICLR*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 95–100. Association for Computational Linguistics.

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

Toan Nguyen and David Chiang. 2018. Improving lexical choice in 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 334–343.

Franz Josef Och and Hermann Ney. 2000. A comparison of alignment models for statistical machine translation. In *COLING 2000 Volume 2: The 18th International Conference on Computational Linguistics*.

Paul Over, Hoa Dang, and Donna Harman. 2007. Duc in context. *Information Processing & Management*, 43(6):1506–1520.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. *ICLR*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163.

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.

Natalie Schluter. 2017. The limits of automatic summarisation according to rouge. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 41–45.

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)*, volume 1, pages 1073–1083.

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

Shiv Shankar, Siddhant Garg, and Sunita Sarawagi. 2018. Surprisingly easy hard-attention for sequence to sequence learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 640–645.

Shiv Shankar and Sunita Sarawagi. 2019. Posterior attention models for sequence to sequence learning. In *International Conference on Learning Representations*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080.

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 *Advances in Neural Information Processing Systems*, pages 5998–6008.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. 2016. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1671–1682.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2017. Reference-aware language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1850–1859.

Lei Yu, Jan Buys, and Phil Blunsom. 2016. Online segment to segment neural transduction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1307–1316.

Fangfang Zhang, Jin-ge Yao, and Rui Yan. 2018. On the abstractiveness of neural document summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 785–790.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1095–1104.