

# Contextual Caption Generation Using Attribute Model

Jessin Donnyson  
School of Electrical Engineering and Informatics  
Institut Teknologi Bandung  
Bandung, Indonesia  
jessinra@gmail.com

Masayu Leylia Khodra  
School of Electrical Engineering and Informatics  
Institut Teknologi Bandung  
Bandung, Indonesia  
masayu@stei.itb.ac.id

**Abstract**— Image captioning has been a big field of research and gained many enterprises' attention in the past few years. Unfortunately, there are still many issues preventing the general usage of image captioning. One of the pitfalls is lack of context. The common approach to solve this issue is to retrain the model with a suitable dataset for every purpose, but this approach is expensive and time-consuming. We propose a method to shift words' distribution and guide the captioning process towards the desired result using a gradient vector acquired from attribute models. This method requires neither retraining captioning models nor customized datasets. In conclusion, this approach shows positive results even though the usage is still limited and highly dependent on the added context and parameter used.

**Keywords**—image captioning, context, attribute models, transformer architecture

## I. INTRODUCTION

Image captioning is a task to automatically produce a sentence that describes the visual content of an image. This task has seen rapid popularity growth in the past few years. Vinyals et al. [1] developed an encoder-decoder architecture for image captioning. Xu et al. [2] successfully showed the power of the sequence-to-sequence approach with an attention mechanism that allows the model to change the subject of focus for each generation step. Many breakthroughs have been achieved ever since then.

Unfortunately, many enterprises still find it difficult to apply image captioning technology in productions. This is generally caused by the high cost of training a custom captioning model or preparing a custom dataset to fit specific needs. Although there are some pre-trained models available for use, controlling caption generations becomes difficult without re-training (or fine-tuning) using specific-dataset or modifying the model architecture to allow extra information to be passed (Keskar et al., 2019) [3]. Both approaches are shown by recent Sharma et al. [4] and Park et al. [5] papers.

An intuitive way to enable controllable caption generation is to model  $p(x|a)$  where  $x$  is the generated caption and  $a$  is some desired attribute. For example, Park et al. [5] trained a model to learn user personal's captions style, and then use this representation to create a personalized caption. In language modeling, Dathathri et al. [6] developed a mechanism for generating conditional text (with various attributes) from pre-trained unconditional language models by utilizing the 'desired attribute'  $a$  in form of bag-of-words (BoW) and simple discriminator (classifier) models. Similarly, Nguyen et al. [7] developed a mechanism for generating images with different characteristics by using an attribute model  $p(a|x)$  together with a standard generative model  $p(x)$ . Using Bayes'

theorem, Nguyen et al. [7] effectively created a conditional generative model  $p(x|a) \propto p(a|x)p(x)$  which is flexible and controllable on the fly.

In a similar way to the solution that Dathathri et al. [6] had proposed for language model, we propose a method to shift words' distribution  $p(x)$  and guide the captioning process towards the desired result  $p(x|a)$  using a gradient vector  $a$  acquired from a bag-of-words (BoW) representing the context to be added. This method allows a generic pre-trained captioning model to incorporate custom 'context' on the fly without the need to be retrained nor fine-tuned using a custom dataset (example in Fig. 1).

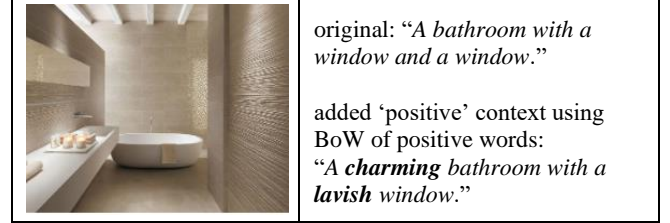


Fig. 1. Example of Adding 'Positive' Context. We change the original caption into a more 'positive' caption by using a bag-of-words filled with positive words.

In addition, recent BERT's breakthroughs in language modeling have inspired us to incorporate transformer architecture for both image captioning's model and word embedding models. We proposed that by using contextual word embedding extracted from those language models (e.g. BERT, GPT-2), the image captioning model should receive a boost in generated captions quality. BERT and GPT-2 are different where BERT is an encoder and GPT-2 is a decoder by nature. This difference in nature might cause great result difference when used as contextual word embedding. Therefore, we decided to test both in this work.

In this paper, we show related works for controllable generation and contextual caption generation in section II. Section III contains the details of the proposed approach. In section IV we show the evaluation of the experiment that has been done. Finally, the conclusion is in section V.

## II. RELATED WORKS

Sharma et al. [4] created a general contextualized image captioning model by constructing a whole new contextualized dataset. This new dataset was named *Conceptual Captions dataset* and have around 15M images. Sharma et al. collected around 5 billion images found on the internet with *alt-text* HTML tag available. Those images went through a lot of pre-processing steps to separate 'usable' images from unusable ones, allowing this dataset to have high-quality contextualized image-captions pairs. After the dataset construction, Sharma

et al. build image captioning models using seq2seq and transformer-based. Both models show improvement when trained with the *Conceptual Captions dataset* compared to the MSCOCO dataset.

Nguyen et al. [7] proposed PPGN (*Plug and Play Generative Networks*). PPGN is made of a generator network that 'draws' a wide range of images and a 'condition' network that tells the generator what to 'draw'. The most beneficial feature of PPGN is the *plug-and-play* which allows one to use a replaceable condition network and generate images according to a condition specified at test time.

Inspired by PPGN, Dathathri et al. [6] proposed PPLM (*Plug and Play Language Model*). This *plug-and-play* mechanism allows pre-trained unconditional language models  $p(x)$  (e.g. GPT-2) to be used as a multi-purpose decoder for generating conditional text  $p(x|a)$ . Meaning that the text generated by this PPLM model could have a certain style, theme, or sentiment which all adjustable and controllable by the user on the fly. Dathathri et al. employed simple bag-of-words to represent style or theme, and a simple classifier model to incorporate sentiments. The result shows that PPLM can generate conditional text on several bag-of-words very well.

Park et al. [5] showed that it's possible to use a real 'information' vector (instead of arbitrary vector) to add personal style to the generated captions. They collect each user's common-used hashtag and convert it into a single vector representation  $h$ . This vector then concatenated into the image's vector and previous generated words' vector to predict the next word distribution. In their paper,  $h$  is defined to be a one-hot vector of top-D most used words for a given user, sorted by TF-IDF scores. This approach technically can produce different styles of captions respective to  $h$  even though the author didn't explicitly say it.

A different approach was applied by Keskar et al. [3] where they prepared several *control-code* upon training a language model. Keskar et al. gathered different styles of text from various domains including Wikipedias, sub-Reddits, OpenWebText, Amazon Reviews, etc. and assign each sample text with *control-code(s)*. The basic idea stays the same where the model is maximizing  $p(x|c)$  where  $c$  is the *control-code*. This method yields amazing results since the model is specifically trained for each *control-code*, but this comes with some cost. Even though users can generate text with different styles by changing the *control-code* used, the flexibility is compromised since the *control-codes* need to be defined from the very beginning. Not to mention the size of the model required is huge (1.6 B parameters).

### III. PROPOSED APPROACH

The overview of the method that we proposed and where the process occurs in generic image captioning's framework can be seen in Fig.2.

#### A. Image Captioning

Image captioning models generally composed of encoder and decoder. An encoder is a part that encodes image into vector representations, while the decoder is the part that decodes the text sequences. This encoder-decoders are trained to compute the conditional probability of sequence  $p(X, I)$  where  $X$  is a sequence of tokens  $X = \{x_0, \dots, x_n\}$ , and  $I$  is the

encoder's hidden state representing images. This probability can be rewritten by applying the chain-rule [10] as:

$$p(X, I) = \prod_{i=1}^n p(x_i | x_0, \dots, x_{i-1}, I) \quad (1)$$

Furthermore, during each step of decoding, both encoder and decoder optionally could incorporate attention mechanism to enable conditional generation specifically for each step  $i$ , improving (1) into:

$$p(X, I) = \prod_{i=1}^n p(x_i | x_0, \dots, x_{i-1}, E_i, D_i) \quad (2)$$

where  $E_i$  is encoder's image representation and  $D_i$  is the decoder's hidden state with attention to step  $i$  respectively. Some merge both  $E_i$  and  $D_i$  as  $H_i$  for simplicity. Tanti et al. [8] suggested two methods to combine image and text representations: *inject* and *merge*. We adopt the *merge* method (i.e. both vectors are processed separately and only merged before the last fully connected layer) throughout the experiments since it's more resource-efficient. We implemented two simple image captioning models using seq2seq [2] and transformer-based using original Vaswani et al. [9] architecture for experiment purposes.

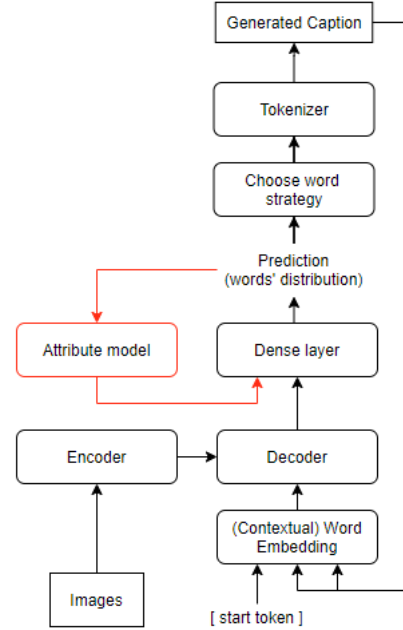


Fig. 2. Overview of Shifting Process (red) in Comparison to Generic Image captioning Framework (black). The red arrow represents the shifting done by the attribute model.

#### B. Shifting distribution

As previously stated, we proposed a method to 'shift' the words' distribution  $p(x)$  and guide the captioning process towards the desired result  $p(x|a)$ . Therefore, we can define a decoder's output function for each token as:

$$\text{(original)} \quad Dec(x_t) = p(x_t | x_1 \dots x_{t-1}, H_t) \quad (3)$$

$$\text{(shifted)} \quad Dec'(x_t) = p(x_t | x_1 \dots x_{t-1}, H_t, a) \quad (4)$$

where  $x$  is the token inside vocabulary,  $H$  represents the decoder's hidden state, and  $a$  is the vector that's going to guide the captioning process by shifting distribution. Overall process can be seen in Fig.3.

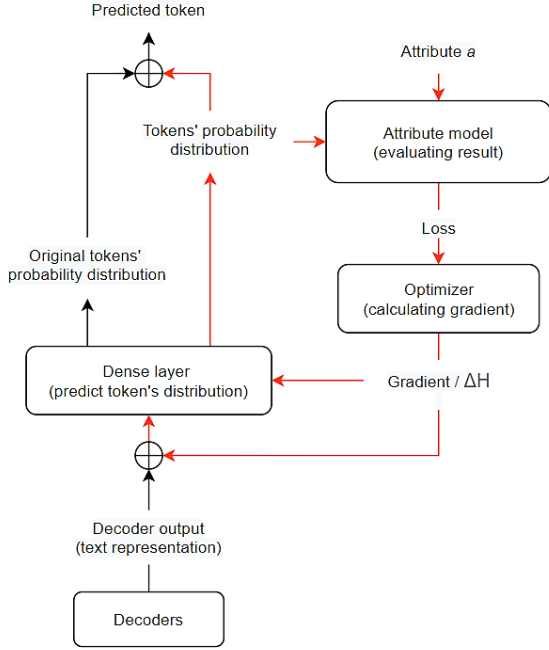


Fig. 3. Illustration of Distribution Shifting. The red arrow represents the shifting mechanism by using the attribute model. For every timestep, we iteratively fed generated tokens' distribution into attribute model, which will output (for each token  $x$ ). Then we compute the loss and the gradient that maximize for the next iteration. We use this gradient to modify current hidden state  $H$ . Finally we use a mixture of original and shifted distribution to output the predicted token.

As seen in (4), to control the output of the captioning model, at every step  $t$ , we shift the output distribution by 'adding' attribute  $a$ . This attribute  $a$  could be informative (representation or arbitrary) vector [3][5], or we could use gradient-based vector [6].

Informative vector tends to be learned together with the captioning model (and optimized during the training), thus resulting in far better captions. Meanwhile, the latter doesn't require the captioning model to be re-trained and can be learned separately, but it has worse results as the captioning model isn't trained nor optimized to maximize  $p(x|a)$ . Using gradient-based vectors can be seen as 'harsh/forced' changes, meanwhile, informative vectors have a more subtle impact since the captioning models already know how to deal with it. Both have their pros and cons.

In this paper, we choose to employ a gradient-based vector [6] (similar to Dathathri et al., 2019) as it allows the captioning model to be flexible and independent regardless of the context or attribute model used. This reason also aligns with the problem of the expensive cost to retrain a captioning model. Flexibility also become one of our main concerns in order to deliver contextual captions, as there's a countless possibility of context to be added into captions.

Adopting the method that's proposed by Dathathri et al., we shift the hidden state  $H_t$  in the direction of two gradients. The first one is the gradient that maximizes unconditioned probability  $p(x)$  (just like what normal image captioning model supposed to do). The other one maximizes the log-likelihood of the attribute  $a$  under the conditional attribute model  $p(a|x)$ . In other words, we shift  $H_t$  so the output of (3) (tokens' probability) will maximize the probability of  $a$  when fed into the attribute model.

Formally, this gradient  $g$  maximizes the probability of token  $x$  so that output of model attribute  $D(x)$  closest to (the value of) attribute  $a$ .

$$g = \operatorname{argmax}_x f(x), \text{ where } f(x) = p(a | D(x)) \quad (5)$$

Like gradient descent, this shifting process can be done for  $n$  times each. We follow Dathathri et al. suggestion of limiting  $n$  value between 3 to 10 and investigate the effect in our experiments. The basic idea is to reduce the magnitude and increase the frequency of updates. This way, it's more likely to reach the maxima since the distribution gradually moves instead of having one big hop towards the target. To illustrate this  $n$  iteration, we denote  $\Delta H_{t,i}$  to be the  $i$ -th update to  $H_t$  during step  $t$  so that the notation for shifted distribution  $H'_t$  as follows:

$$H'_t = H_t + \sum_{i=0}^n \Delta H_{t,i} \text{ where } \Delta H_{t,0} = 0 \quad (6)$$

Generally,  $\Delta H_t$  is initialized with zeros for the very first calculation for each timestep since we require the unconditioned-original probability of  $x$  to calculate the gradients (5). Furthermore, we can expand  $\Delta H_{t,i}$  as [6]:

$$\Delta H_{t,i} = \Delta H_{t,i-1} + \alpha \frac{d(\log p(a | x_t))}{dx} \quad (7)$$

Equation (7) also equivalent to the definition denoted by Dathathri et al. [6] where they rewrite the attribute model  $p(a | x)$  as  $p(a | H_t + \Delta H_t)$ :

$$\Delta H_t \leftarrow \Delta H_t + \alpha \frac{d(\log p(a | H_t + \Delta H_t))}{d\Delta H_t} \quad (8)$$

where  $\alpha$  is the adjustable update's weight (or so-called step size).

The reason behind the usage of these two gradients at the same time is to enable a 'controllable force' when shifting the distribution. Meaning users can apply 'full-force shift' by using only the second gradient, or 'moderate-force shift' by using the weighted average of the first and second gradient. This 'controllable force' can also be done by applying a weighted average for shifted distribution and original distribution. The latter has a more noticeable impact.

Note that the 'shift' is restricted to  $H_t$ , meaning each 'shift' is calculated solely for that timestep. Note that  $H_t$  still can affect  $H_{t+1}$  by successfully changing predicted token  $x_t$  to  $x'_t$  (since  $H_{t+1}$  will take  $x'_t$  into account), but will not affect  $H_{t+1}$  if failed to changes or the shift is not impactful enough to be done.

### C. Controlling shift

As mentioned earlier, one's can control the power of shift by directly manipulating the weighted average ratio of the original distribution and shifted distribution. For example, we can rewrite shifted tokens' probability (4) as:

$$Dec'(x_t) = (1 - \gamma) P_x + \gamma P'_x \quad (9)$$

$$(\text{original}) P_x = p(x_t | x_1 \dots x_{t-1}, H_t)$$

$$(\text{shifted}) P'_x = p(x_t | x_1 \dots x_{t-1}, H'_t)$$

we shall call this ratio  $\gamma$  as *gamma* to simplify things. Gamma becomes one of the important parameters to tune as it directly controls how much influence the shifted distribution has over the original distribution.

As stated earlier, in the case of using a gradient vector, the captioning models are not trained to maximize  $p(x_t | a)$  for attribute  $a$  during the initial training of image captioning. Thus, choosing the right value for each shifting parameter becomes more important. We identified four possible outcomes:

1. Best-case: when the updated distribution generates a better caption and possess the desired attribute.
2. Overwhelming: when the update is going in the right direction, but the magnitude is too large.
3. Underwhelming: when the update is going in the right direction, but the magnitude is too small.
4. Worst-case: when the updated distribution fails to generate sounds captions, throwing random irrelevant words instead.

To ensure the quality of captions, we introduce four hyperparameters used to control the result:

1. Iteration ( $n$ ): number-of-times shifting occurs for each timestep
2. Update weight ( $\alpha$ ): weighting used to update the hidden state  $H_t$
3. Gamma ( $\gamma$ ): the ratio of shifted distribution and original distribution
4. Optimizer's Lr: Optimizer's internal learning rate used to compute the gradient.

during experiment, we found that using  $n = 3 - 5$ ,  $\gamma = 0.8 - 0.9$ , and Optimizer's Lr = 0.005 – 0.01 yield better result. As for  $\alpha$ , we find that it's less significant. We sorted most significant to least significant out of these four: *gamma* ( $\gamma$ ), optimizer's lr, iteration ( $n$ ), and update weight ( $\alpha$ )

#### IV. EXPERIMENT AND EVALUATION

During the experiment, we use custom vocabulary mapping to reduce the vocab size of BERT and GPT-2. We select 5000 tokens with the highest occurrence count, details on Table I.

TABLE I. TOKEN MAPPING COVERAGE

Dataset	Tokenizer / word embedding	Unique token coverage	Token coverage
MSCOCO	GPT-2	24.41 %	98.29 %
	BERT	34.39 %	99.20 %
	Keras	21.11%	98.69%
Flickr 8k	GPT-2	58.66 %	99.12 %
	BERT	67.35 %	99.54 %
	Keras	58.85%	99.17%

We train both of our captioning models with both Flickr 8k and MSCOCO dataset. We separate 20% of the images from each dataset for test purpose. We further split the rest 80% with 80:20 ratio between train and eval after shuffling image-captions pairs.

##### A. Experiment Scenario

There are two parts of the experiment that we conducted in this work. The first part (P1) is to re-create image captioning models and examine the effect of using contextual word embedding extracted from language models such as BERT and GPT-2 compared to standard Keras embedding. The second part (P2) is to add context to the generated captions using the method that we proposed earlier.

For P1, we create simple seq2seq and transformer-based models trained them until the model converges. We use LSTM as the recurrent network for the seq2seq model, with specification: 768 units, dropout 0.1, batch size 64, 12-15 epoch. Meanwhile the specification for transformer model: 6 decoder layers, 768 units (split into 8 heads), batch-norm & dropout 0.1, batch size 64, 8-12 epochs. Both models went through minor fine-tuning as our goal is not to surpass *state-of-the-art* of the dataset. We use 768 units for both models to comply with BERT's and GPT-2's hidden state/embedding size.

For P2, we use bag-of-words as our attribute model of choice for its simplicity, and we use the same bag of words used by Dathathri et al. [6]. We first find 'best' parameters by using automated metrics, then finally evaluate two of the 'best' by human review (subjective metrics). Note that there's no common automated metric for measuring 'how well a context being added', therefore we suggest using these four metrics (all metrics higher means better):

1. BLEU: A standard metric for text similarity, also commonly used in image captioning.
2. delta BLEU (dB): changes of BLEU before and after being added context.
3. delta Occurrence (dOc): changes of the occurrence of tokens inside BoW, before and after added context; divided by the number of captions generated. This equivalent to the expectation of a single word inside BoW to be added into the caption.
4. delta Recall (dRc): changes of the ratio of tokens inside BoW and tokens outside BoW for each caption.

For subjective metrics, we use three aspects which are: fluency (Fl), caption's relevance (Cr), and context quality (Ctx) with score range up to 4. Review's guideline is available in the <sup>1</sup> GitHub repository.

##### B. Experiment Results

Table II shows the result for experiment P1. Based on the experiment, we conclude that usage of GPT-2 as contextual word embedding always leads to a better result (noticeable boost compared to runner-up) for both for seq2seq or transformer models. This finding justifies our hypothesis about the benefits of using a contextual word embedding (from language models) instead of a standard embedding layer. On the other hand, the Transformer-BERT model shows a surprisingly disappointing result of BLEU 0.044.

<sup>1</sup> GitHub: [github.com/Jessinra/Papers](https://github.com/Jessinra/Papers)



TABLE II. P1'S EXPERIMENT RESULT

Model	BLEU-1	BLEU-4	METEOR	ROUGE-L
LSTM- Keras	0.572	0.159	0.191	0.380
LSTM- BERT	0.536	0.134	0.199	0.373
LSTM- GPT-2	0.611	0.169	0.207	0.431
Transformer- Keras	0.581	0.164	0.196	0.382
Transformer- BERT	0.044	0.000	0.022	0.055
Transformer- BERT*	0.500	0.113	0.179	0.349
<b>Transformer- GPT-2</b>	<b>0.619</b>	<b>0.187</b>	<b>0.212</b>	<b>0.438</b>

We investigated further and found that the nature of BERT itself is causing this issue. BERT by nature is a bidirectional encoder and not suitable for auto-regression tasks like image captioning, especially when we didn't train the model sequentially. For every timestep, BERT assumes the sentence is already completed, which led to inconsistent representation. Meaning, first  $n$ -word in a sentence of length  $N$ , will have different representations when the sentence has only  $n$  length. The transformer model wasn't trained to predict using these 'partial' representations, which led to a bad outcome during caption generation. Hence, we split the training captions into  $n$  timesteps to address this issue (like seq2seq training style) and successfully trained transformer-BERT\*.

TABLE III. P2'S AUTOMATED EVALUATION RESULT

Parameters	B $\uparrow$	dB% $\uparrow$	dOc% $\uparrow$	dRc% $\uparrow$
iter 3 lr 0.05 optlr 0.03 gamma 0.8	<b>0.579</b>	-3.33	36.0	-15.16
iter 5 lr 0.005 optlr 0.01 gamma 0.8	0.560	-6.51	48.0	<b>174.99</b>
<b>iter 5 lr 0.05 optlr 0.01 gamma 0.8</b>	<b>0.576</b>	<b>1.45</b>	<b>60.0</b>	45.1
<b>iter 5 lr 0.3 optlr 0.01 gamma 0.8</b>	0.562	<b>-0.88</b>	<b>60.0</b>	<b>232.22</b>
iter 5 lr 0.5 optlr 0.01 gamma 0.8	0.534	-3.18	36.0	169.78

B = Bleu-1, dB% = delta Bleu (%), dOc% = delta Occurrence (%), dRc% = delta Recall (%), highlighted top 2 for each metric.

Table III is a compact version of the result for experiment P2, the full version is in the <sup>1</sup>GitHub repository. The sample of adding context results are available in Table V. We used greedy-grid-search to find 'best' parameters by pruning candidates iteratively based on the metrics evaluated on (started at zero and increase little by little until only  $k$  top candidate left). Increased value of any of those 4 parameters tends to lower BLEU-related scores (B, dB) and increase scores related to context occurrence (dOc, dRc). Finding the 'best' parameters was not easy and it heavily affects the result of context added. The real challenge is to find the right balance between BLEU drop and increased occurrence rate.

Table IV shows the human evaluation result towards 175 image-captions pairs for each model, which contain mixes of four different bag-of-words used.

TABLE IV. P2'S HUMAN EVALUATION RESULT

iter 5 lr 0.05 optlr 0.01 gamma 0.8					
Bag of Words	Original caption		Added context		
	Fl	Cr	Fl	Cr	Ctx
Military	3.34	3.24	3.04	2.40	2.88
Religion	3.12	2.88	3.15	2.4	2.72
Legal	3.33	3.23	3.18	2.63	2.70
Positive Words	3.26	2.86	2.91	2.66	2.66
iter 5 lr 0.3 optlr 0.01 gamma 0.8					
Bag of Words	Original caption		Added context		
	Fl	Cr	Fl	Cr	Ctx
Military	3.12	3.04	3.04	2.37	2.89
Religion	3.40	2.85	3.20	2.35	2.72
Legal	3.23	3.23	2.95	2.65	2.60
Positive Words	3.43	3.14	2.89	2.66	2.71





Based on experiments, adding context tends to reduce the fluency and caption relevance. The fluency scores (Fl) range between 2.9 to 3.1 after being added context, reduced by about 0.1 to 0.3. This shows that even after we shift the original distribution, the model still capable of producing grammatically correct captions.

Heuristically we could predict that the caption relevance score (Cr) will drop after we added context due to the 'forced' distribution shift. Table IV shows that Cr dropped about 0.5 on average (half index). Meaning previously caption was almost perfect, after being added context it becomes somewhat correct. We believe this happened because the model is 'forced' to incorporate 'unnecessary' tokens where it doesn't belong initially. We also didn't consider the suitability between the caption and the context.

The context quality score (Ctx) achieved by models range between 2.6 – 2.8. This means the model can select and incorporate suitable token inside the bag-of-words into the caption. The usage of the context is also relatively good, even though most of the time, it doesn't necessarily show in the image. Once again, we believe this happened because we didn't specifically use suitable bag-of-words for each image. Therefore, the context added doesn't show in the image.

From four bag-of-words that we tested, we found that specific BoW (military, religion, legal) usage is easier than general BoW (positive words). We believe that this is caused by a large number of tokens inside positive-words BoW (800+ token). For BoW attribute models, a large number of tokens makes the distribution's shift become more evenly done. Which eventually 'confuse' the captioning model as there are too many tokens probability to be maximized. It'll also reduce Fl and Cr since the shift is much more significant (it becomes harder to maintain fluency and relevance). We suggest using a BoW that contains around 50 – 70 tokens.

TABLE V. ADDING CONTEXT SAMPLES

Best-case		original: A bathroom with a window and a window.  added context (BoW positive words): A <b>charming</b> bathroom with a <b>lavish</b> window
Overwhelming		original: A group of chefs preparing food in a kitchen.  added context (BoW positive words): A group of people <b>entertaining a fabulous</b> woman in a kitchen
Underwhelming		original: A bird sitting on a wooden bench.  added context (BoW positive words): A bird sitting on a wooden bench.
Worst-case		original: A plane is parked on the runway.  added context (BoW religion): A Suinate Retral planeer Retpect Graorthvert Canon YearCom ForceResumblelife Chavenespherdquest.

## V. CONCLUSION

We proposed a method to add context to the generated captions by using attribute models. By using gradient-based vectors, the captioning model requires no re-training process nor custom dataset. This method allows users to add any kind of context as long it's representable by an attribute model. The distribution shifting process can be controlled by modifying several hyper-parameters which allow the user to adjust the power on the fly and further improves the model flexibility.

In our experiment, we successfully employed several bag-of-words as attribute models to demonstrate the method we proposed. The experiment shows positive results even though the quality of the context added depends on the image's caption, context-image relevance, and the hyper-parameters used. We also show that by using contextual word embedding extracted from a certain language model, the captioning model will achieve a better result.

For future works, we suggest exploring different types of attribute models. Let it be sentiment-classifier, tone-classifier, domain-classifier, or any other models you have, to further increase the flexibility of the captioning model.

## VI. REFERENCES

- [1] Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). Show and tell: A neural image caption generator. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3156-3164).
- [2] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., ... & Bengio, Y. (2015, June). Show, attend and tell: Neural image caption generation with visual attention. In International conference on machine learning (pp. 2048-2057).
- [3] Keskar, N. S., McCann, B., Varshney, L. R., Xiong, C., & Socher, R. (2019). Ctrl: A conditional transformer language model for controllable generation. arXiv preprint arXiv:1909.05858.
- [4] Sharma, P., Ding, N., Goodman, S., & Soricut, R. (2018, July). Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 2556-2565).
- [5] Chunseong Park, C., Kim, B., & Kim, G. (2017). Attend to you: Personalized image captioning with context sequence memory networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 895-903).
- [6] Dathathri, S., Madotto, A., Lan, J., Hung, J., Frank, E., Molino, P., ... & Liu, R. (2019). Plug and play language models: a simple approach to controlled text generation. arXiv preprint arXiv:1912.02164.
- [7] Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. Plug & Play Generative Networks: Conditional Iterative Generation of Images in Latent Space. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017.
- [8] Tanti, M., Gatt, A., & Camilleri, K. P. (2018). Where to put the image in an image caption generator. Natural Language Engineering, 24(3), 467-489.
- [9] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).
- [10] Manning, C. D., Manning, C. D., & Schütze, H. (1999). Foundations of statistical natural language processing. MIT press.