# Conditional Text Generation: generations of topic-aware captions

Matteo Bonina
Politecnico di Torino
s267308@studenti.polito.it

Raffaella Ciancio
Politecnico di Torino
s270426@studenti.polito.it

Benedetta Giorgi
Politecnico di Torino
s270004@studenti.polito.it

Alessandro La Rocca
Politecnico di Torino
s269777@studenti.polito.it

## Abstract

*The possibility of having control on the specific generated text is a crucial element in text generation models. In order to achieve this result, we fine-tuned CTRL model by Salesforce, which is able to generate conditional text considering several factors such as context, topic or emotion. We trained CTRL model on images' captions of COCO dataset in order to be able to generate context well-related sentences starting from a word, called control code, that summarizes a concept. Other several specific experiments were hold to test the accuracy of our model and to better evaluate the choices of CTRL generation's parameters. All our experiments are available at https://github.com/AlessandroLaRocca96/LinksTextGen.*

## 1. Introduction

Text generation is a discipline that has become one major area of Research and Development in natural language processing.
Its central aim is to automatically generate natural language texts, which can satisfy several communicative requirements, in order to appear indistinguishable with respect to text written by humans.
Nowadays, this has become one of the most important challenging tasks, in particular with the exponential growth of the neural networks.
In this paper, we mainly focus on conditional text generation (CTG) and in particular on topic-aware text generation. Compared with general text generation, conditional text generation is more in line with the needs of precision and friendly services.

Starting from CTRL model developed by Salesforce [6], our work aimed to perform fine-tuning of this model and draw conclusions about human understandability of the generated text. The work is divided in two phases:

- Fine-tuning of the CTRL model using the captions retrieved from the COCO dataset.
- Quality evaluation on the generated text performing various experiments.

For the former point, we have splitted the dataset into 12 supercategories describing the sentiment of the images. Then we have fine-tuned our model in order to generate captions related to domain of the specific supercategory. In this way the model is able to generate not only sentences in a general way but sentences that are caption-like.

For the latter point, the work is based on computing some metrics, in particular BLEU, Self-BLEU and POS-BLEU, that allow us to evaluate text complexity, variety and how much a sentence is human understandable.

## 2. Related work

### 2.1. Conditional text generation

Conditional text generation is an important task of text generation, aiming to generate realistic text that carries a specific attribute (e.g., positive or negative sentiment) [8]. The main objective is to take into consideration external conditions in order to influence the generated results in the process of text generation. These conditions usually include context, topic, emotion, external knowledge, and so on. The general text generation methods consider only the text content, which makes the generated text less diverse and not comparable to human expression. Consideration of external conditions in text generation makes it more anthropomorphic and brings better services to human beings in various fields. [1]

### 2.2. CTRL - A Conditional Transformer Language Model for Controllable Generation

The model used for our experiments is CTRL [6], a 1.63 billion-parameter conditional transformer language

model, trained to condition on special keywords called control codes that govern style, content, and task-specific behaviour.

## 2.3. How it works

CTRL is a conditional language model that is always conditioned on a control code *c* and learns the distribution *p( x | c )* . The distribution can still be decomposed using the chain rule of probability and trained with a loss that takes the control code into account.

$$p(x|c) = \prod_{i=1}^{n} p(x_i|x_{<i}, c)$$

The control code *c* provides a point of control over the generation process. CTRL learns $p_\theta(x_i|x_{<i}, c)$ by training on sequences of raw text prepended with control codes [6].
The training is based on the concept of tokenization .
Tokenizing a text means splitting it into words or subwords, which are then converted to IDs through a look-up table. Data is tokenized through fastBPE [3], with a vocabulary of 250k tokens. This includes the sub-word tokens necessary to mitigate problems with rare words, but it also reduces the average number of tokens required to generate long text by including most common words. Each sequence is originated from a domain, and it has the corresponding domain control code prepended as the first token in the sequence. In this way, domain control codes receive special treatment: they are propagated to all text in the domain as the first token.

## 2.4. Text generation

After training, the model learns to generate a conditional probability distribution over the vocabulary of tokens according to the given input sequence, which has to be a control code in order to obtain a good result.
During the generation process, several parameters can be modified in order to obtain different probability distributions and, therefore, try to influence the way to select the next token from this distribution.
They are Temperature, Top-K, Nucleus and Penalty.

### 2.4.1 Temperature

Temperature is a hyperparameter of neural networks used to control the randomness of predictions by scaling the logits before applying softmax. It is a scaling factor that tells how the model is conservative in the generation of text, for example using easy and safe words instead of complex words.
When temperature is equal to 1, the softmax function is computed directly on the unscaled output of earlier layers, that is the logits.
Instead, with a value of temperature different from 1, the softmax is computed on ratio between logits and temperature.
If the value of temperature is lower than 1, the result will be larger and it makes the model more confident but also more conservative in its samples.
If the value of temperature is higher than 1, the probability distribution will be softer over the classes, and the text generated will be more diversified and with a lot of mistakes.
In general, given a temperature *T>0* and scores $x_i$, for each token *i* in the vocabulary, the probability of predicting the *i-th* token is given by:

$$p_i = \frac{\exp(x_i/T)}{\sum_j \exp(x_j/T)}$$

The next token is then chosen by sampling through a multinomial distribution with probabilities $p_i$ clipped at the top-k tokens [6]. Increasing the temperature is a simple method for trying to encourage more diversity in decoded outputs. A good trade-off (the standard value proposed by CTRL Salesforce) seems to be *t=0.5* because it makes the model more robust to errors while maintaining some diversity.

### 2.4.2 Top-K

In its most basic form, sampling means randomly picking the next word $w_t$ according to its conditional probability distribution:

$$w_t = P(w|w_{1:t-1})$$

The Top-K sampling was introduced by Fan et. al (2018).
In Top-K sampling, the *K* most likely next words are filtered, defining a new set $V_{topK}$ and the probability mass is redistributed among only those *K* next words.
At each step of sampling we calculate:

$$\sum_{w \in V_{topK}} P(w|w_{1:t-1})$$

That represents the probability mass in each step.
*w* is the next word candidate and $V_{topK}$ is the space of *K* possible candidates.[7]

### 2.4.3 Nucleus

Limiting the sample pool to a fixed number *K* could lead the model to generate no significant text for sharp distributions and could limit the model's inventiveness for flat distributions.
This problem could be solved using Top-p or Nucleus-Sampling approach. [7]
The basic idea of Nucleus Sampling is that most of the probability mass is concentrated in a small set of the vocabulary.

Text is sampled from the nucleus of the probability distribution which is dynamic and contains the vast majority of the probability mass.
This approach helps us truncating the less important tail of the distribution, adapting the candidate pool dynamically and ends up in fluent, coherent and well diversified text.

#### 2.4.4 Penalty

Penalty is a parameter used to prevent repetitions. It can be used to penalize words that were already generated or belong to the context.[7]
Penalizing the sampling means discounting the scores of previously generated tokens. Penalized sampling is not used during training, but only during the generation process.
We find that using a greedy sampling and $penalty \approx 1.2$ yields a good balance between truthful generation and lack of repetition. [6]

## 3. Our work

The main purpose of our work was to fine-tune CTRL model by Salesforce described above, using COCO dataset's supercategories as control codes.
COCO dataset is a large-scale object detection, segmentation, and captioning dataset. There are different versions of this dataset: we decided to use the 2017 Train/Val annotations [2] which contains captions describing images.
Every caption is associated to a specific supercategory which is a word that represents a certain topic.
We divided our training dataset in other twelve subset datasets, each of these containing captions related to one supercategory.
The same thing was made also on the *Val* dataset which we used as test dataset.
These operations led to twelve training datasets of different dimensions (12k-40k captions) and to twelve test datasets of different dimensions (400-2k captions).
Starting from this, we choose Animals, Appliance, Food, Furniture, Outdoor, Person, Sport datasets among these, which are the most significant for our experiments in terms of size and topic.
After that, we trained our model on all the seven subset training datasets in order to generate different conditional text.
To perform this high computational training, we used an Azure NC6_Promo virtual machine, with 56GB ram, 340Gb HDD, 1 K80 GPU, with a pretrained version of CTRL model on it. We generated captions for each topic mentioned before, expecting meaningful and well context-related sentences.
The results that we obtained were remarkable: we were able to appreciate from the beginning what conditional text generation means.

Here we report some meaningful sentences that show how generation of text changes starting from different control codes but with identical initial prompts:

> `Food` *A man standing in front of a large pile of bananas.*
> `Appliance` *A man standing in front of an open refrigerator looking into the fridge.*

Our model was trained at least for one epoch for every subset training dataset related to every supercategory, then other experiments were hold in order to see if some improvements could have been obtained.
The quality of the text-generated was analyzed computing some metrics evalutations.
All the experiments that required less amount of computation capability were performed using Google Colab.

## 4. Metrics

In the Natural Language Processing (NLP) an automatic way to evaluate the performances of the machine tranlastion systems is the usage of the BLEU score proposed by Kishore Papineni, et. al (2002). [4]
BLEU stands for **Bi**-**Lin**gual **E**valuation **U**nderstudy and it is also widely used to evaluate words similarity between two sentences or documents. The approach works by counting matching n-grams in the candidate text to n-grams in the reference text.

### 4.1. BLEU

BLEU works by computing the precision, i.e. the fraction of tokens in the candidate which are present also in the references. The BLEU score is defined in a range between 0 (worst) and 1 (best).
In the following table it is possible to see a general interpretation of a given BLEU score.

| BLEU score interpretation | |
|---|---|
| **BLEU Score** | **Interpretation** |
| <0.10 | Almost useless |
| 0.10 - 0.19 | Hard to get the gist |
| 0.20 - 0.29 | The gist is clear, but has a significant grammatical errors |
| 0.30 - 0.40 | Understandable to good translations |
| 0.40 - 0.50 | High quality translations |
| 0.50 - 0.60 | Very high quality, adequate, and fluent translations |
| >0.60 | Quality often better than human |

Table 1

A brevity penalty is used as a counterweight to avoid

high scores for short candidates:

$$BP = min(1, \frac{candidate\ length}{reference\ length})$$

### 4.2. Self-BLEU

Self-BLEU is a metric to evaluate the diversity of the considered text. In this kind of approach the reference text used is the candidate text itself. Self-BLEU uses one sentence as candidate and the others as references, computes the BLEU score and then computes an average.

### 4.3. POS-BLEU

POS stands for Part-Of-Speech.
Every word can be associated with a specific tag, called POS-Tag that defines, for each word, if it is a noun, a verb, an adjective, an adverb, etc.
POS-BLEU consists in replacing words contained in the candidate and reference text with the related POS tags.
The BLEU score is then computed between these POS tags instead of words.
POS-BLEU is used to measure the structure similarity between the candidate sentences and the reference ones. It is clear that if the structure of generated text is caption-like, the value of POS-BLEU will be, reasonable, higher.

### 4.4. N-gram language model

BLEU score can be calculated using different n-grams. An n-gram language model predicts the probability of a given n-gram within any sequence of words in the language. Having a good n-gram model, we can predict *p(w | h)*, that is the probability of seeing the word *w* given the history of previous words *h*, where the history contains *n - 1* words.
In particular, BLEU compares the n-gram of the candidate translation with n-gram of the reference translation to count the number of matches. These matches are independent with respect to the positions where they occur.
It is clear that as the value of *n* increases, the value of BLEU tends to decrease.
The more the number of matches between candidate and reference translation, the better is the machine translation.

## 5. Experiments and results

After fine-tuning our model with new datasets, we generated captions in different conditions and analyzed the results through the metrics described above.
BLEU, Self-BLEU and POS-BLEU were computed using different values of n-gram (2,3,4,5). Also, the metrics were calculated using, as reference, 2 different datasets:
-The entire training set
-The test set
In the following, we have made a lot of experiments in different conditions, with the aim of avoiding results due to a particular anomalous condition. The generation has been done with default parameters' values proposed by CTRL:

- Temperature: 0.5
- Top-K: 0
- Nucleus: 0
- Penalty: 1.2

### 5.1. Experiments varying dimension of training dataset

First of all we generated samples of approximately 200 captions using different dimensions datasets. In particular we trained datasets Forniture.txt (40k captions), Animal.txt (26k captions) and Appliance.txt (12k captions) for one epoch. The following table shows the results obtained using the training set as reference text :

| BLEU scores varying training dataset dimension | | | |
|---|---|---|---|
| **BLEU type** | **12k** | **26k** | **40k** |
| BLEU-2 | 0.87 | 0.94 | 0.94 |
| BLEU-3 | 0.69 | 0.77 | 0.81 |
| BLEU-4 | 0.49 | 0.57 | 0.63 |
| BLEU-5 | 0.35 | 0.41 | 0.47 |
| S-BLEU-2 | 0.66 | 0.63 | 0.58 |
| S-BLEU-3 | 0.44 | 0.38 | 0.37 |
| S-BLEU-4 | 0.28 | 0.22 | 0.23 |
| S-BLEU-5 | 0.18 | 0.14 | 0.15 |
| P-BLEU-2 | 0.99 | 0.99 | 0.99 |
| P-BLEU-3 | 0.98 | 0.99 | 0.98 |
| P-BLEU-4 | 0.96 | 0.98 | 0.97 |
| P-BLEU-5 | 0.90 | 0.94 | 0.95 |

Table 2

The following table shows the results obtained using a test set provided by COCO as reference text:

| BLEU scores varying training dataset dimension | | | |
|---|---|---|---|
| **BLEU type** | **12k** | **26k** | **40k** |
| BLEU-2 | 0.54 | 0.69 | 0.71 |
| BLEU-3 | 0.30 | 0.42 | 0.44 |
| BLEU-4 | 0.17 | 0.25 | 0.27 |
| BLEU-5 | 0.11 | 0.16 | 0.18 |
| S-BLEU-2 | 0.66 | 0.63 | 0.58 |
| S-BLEU-3 | 0.44 | 0.38 | 0.37 |
| S-BLEU-4 | 0.28 | 0.22 | 0.23 |
| S-BLEU-5 | 0.18 | 0.14 | 0.15 |
| P-BLEU-2 | 0.97 | 0.99 | 0.98 |
| P-BLEU-3 | 0.91 | 0.95 | 0.96 |
| P-BLEU-4 | 0.81 | 0.86 | 0.90 |
| P-BLEU-5 | 0.67 | 0.73 | 0.80 |

Table 3

As shown in the tables above, BLEU scores get higher as we increase the dimension of the training datasets. That is because our training dataset, which is also our reference text, is bigger, so it is easier to find a match between candidate and reference. However the values obtained for each configuration are very good.
Self-BLEU, instead, gets higher when we train the model on a smaller dataset (like the Appliance.txt one). Our hypothesis is that performing training on a smaller dataset leads to generate sentences based on a smaller vocabulary, so Self-BLEU increases due to the captions' similarity.

### 5.2. Experiments varying number of epochs

The second experiment that we performed was generating captions changing the number of epochs during training phase, in particular with 1, 2 and 10 epochs.
Fine-tuning in this experiment was performed using the Outdoor.txt dataset retrieved from COCO, that contains approximately 14k captions.
The following table shows the results obtained using the training set as reference text:

| BLEU score varing the number of epochs | | | |
|---|---|---|---|
| **BLEU type** | **1 epoch** | **2 epochs** | **10 epochs** |
| BLEU-2 | 0.89 | 0.89 | 0.89 |
| BLEU-3 | 0.75 | 0.75 | 0.77 |
| BLEU-4 | 0.58 | 0.58 | 0.63 |
| BLEU-5 | 0.43 | 0.43 | 0.51 |
| S-BLEU-2 | 0.60 | 0.59 | 0.48 |
| S-BLEU-3 | 0.41 | 0.36 | 0.25 |
| S-BLEU-4 | 0.28 | 0.23 | 0.15 |
| S-BLEU-5 | 0.21 | 0.17 | 0.10 |
| P-BLEU-2 | 0.99 | 0.99 | 0.99 |
| P-BLEU-3 | 0.98 | 0.98 | 0.99 |
| P-BLEU-4 | 0.96 | 0.95 | 0.96 |
| P-BLEU-5 | 0.91 | 0.89 | 0.92 |

Table 4

The following table shows the results obtained using a test set provided by COCO as reference text:

| BLEU score varing the number of epochs | | | |
|---|---|---|---|
| **BLEU type** | **1 epoch** | **2 epochs** | **10 epochs** |
| BLEU-2 | 0.60 | 0.60 | 0.57 |
| BLEU-3 | 0.36 | 0.36 | 0.35 |
| BLEU-4 | 0.22 | 0.21 | 0.22 |
| BLEU-5 | 0.14 | 0.14 | 0.14 |
| S-BLEU-2 | 0.60 | 0.59 | 0.48 |
| S-BLEU-3 | 0.41 | 0.36 | 0.25 |
| S-BLEU-4 | 0.28 | 0.23 | 0.15 |
| S-BLEU-5 | 0.21 | 0.17 | 0.10 |
| P-BLEU-2 | 0.97 | 0.97 | 0.97 |
| P-BLEU-3 | 0.91 | 0.91 | 0.92 |
| P-BLEU-4 | 0.82 | 0.82 | 0.84 |
| P-BLEU-5 | 0.69 | 0.70 | 0.79 |

Table 5

As the tables show, BLEU scores are similar among the models trained with different epochs.
Self-BLEU, instead, decreases. In fact, if we look at the generated captions in a qualitative way, the structure of the generated captions seems to be more complex.
Furthermore, we made a quantitative evaluation which confirmed our first impression that the average length of generated captions is higher in the 10-epochs fine-tuned model.
In the following, there are two different captions generated with the same prompt but different models (1 epoch vs 10 epochs) that clearly show the difference:

`Outdoor` *A street A stop sign posted above a highway.*
`Outdoor` *A street A stop sign, pedestrian lights, and no parking signs on a pole beside a road.*

### 5.3. Experiments varying number of generated captions

Another experiment that we made is to generate even more sentences than the ones previously generated in order to see if there are some remarkable improvements.
We choose to generate 1k captions for Furniture and 1k captions for Person control codes and to calculate BLEU, Self-BLEU and POS-BLEU on them, using as references their corresponding train and test datasets that we used before.
We noticed that nothing changed concerning BLEU and POS-BLEU.
Self-BLEU, instead, as it is possible to see in the figure below, was slightly increasing due to the fact that the number of references is 1k instead of 200, so it is more likely that a sentence can be found multiple times in the same generated text.

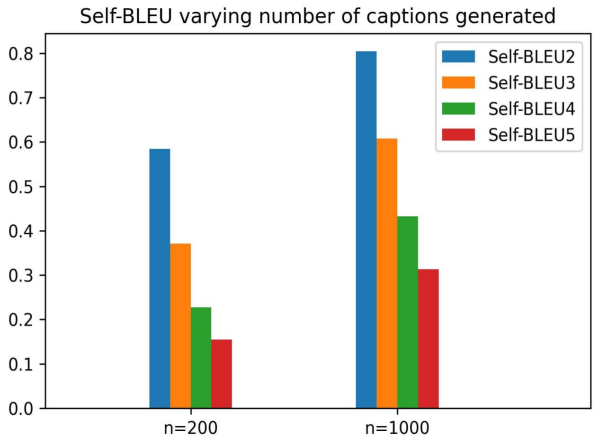


Figure 1

#### 5.3.1 Other considerations

In addition to the metrics proposed before, we decided to make additional considerations on our generated captions.
We checked if every of the generated sentences, in any case, is somehow identical to a sentence in its corresponding training dataset. We noticed that this did not happen except in few cases, where one or two captions at most were repeated, meaning that our model is well trained.

### 5.4. Experiments varying the parameters

The generation process can be highly influenced by the values of Temperature, Top-K, Nucleus and Penalty.
As said before, our previous experiments were carried out with the standard values proposed by CTRL in order to obtain texts which are not repetitive or inconsistent, but sufficiently diversified.
With the aim of analyzing as much as possible the quality of the generated text, we decided to make one more experiment to evaluate how decisive the choice of these parameters, actually, is.
For each of the parameters, we have chosen 3 values as different as possible, because the choice of similar values would probably have led to almost the same results.
Our analysis is based on generating 200 captions for each configuration and computing Self-BLEU as a metric of diversity. We decided to evaluate this metric because the definition itself of Self-BLEU makes us understand that a lower Self-BLEU score implies higher diversity in generated text.

#### 5.4.1 Temperature

The experiment was made with *t=0.1, t=0.5, t=1*. Increasing the value of temperature, the text generated will be more diversified, with a lower Self-BLEU score consequently. On the other hand, a lower value of temperature makes the generation more conservative, with fewer risky words and a higher Self-BLEU score. In the following plot it is possible to see the trend:

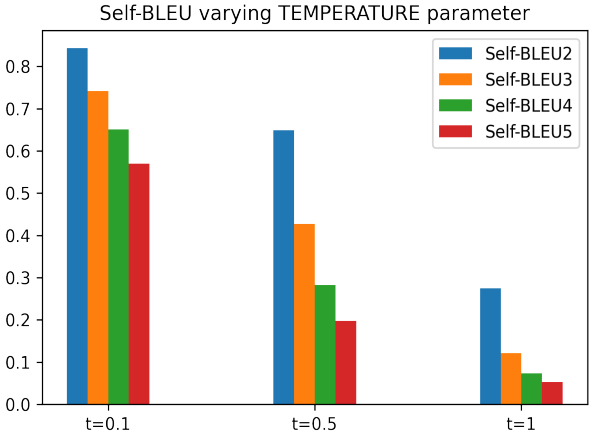


Figure 2

#### 5.4.2 Top-K

The generation of captions can be affected by Top-K parameter.
In our experiment we have considered as values of *k : 15, 40, 150*.
Values greater than 150 did not show relevant results. In the following plot it is possible to see the trend of Self-BLEU:

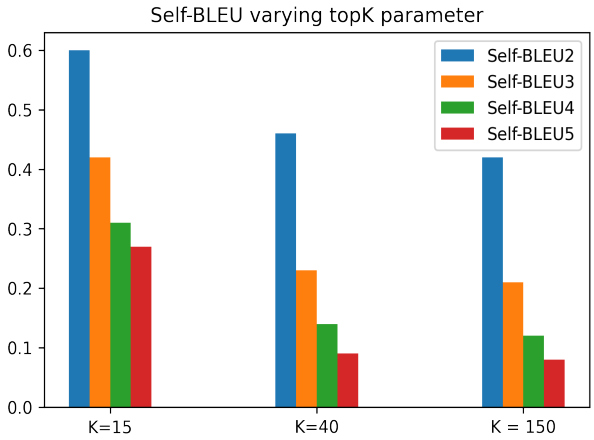


Figure 3

We have a decreasing trend of Self-BLEU score. This is due to the fact that with an higher K value, we filter the K most likely next words and redistribute the probability mass on these ones. Note that the difference of Self-BLEU score between *K = 15* and *K = 40* is evident. For values of *K >40* the trend keeps steady.

#### 5.4.3 Nucleus

As far as Nucleus parameter variation is concerned, we made experiments with *p=0.1, p=0.95, p=0.99*, obtaining a similar trend to the one obtained with the other parameters. Increasing $p$ parameter means expanding the pool of words from which our model can pick new words during generations.
This ends up in higher diversity in captions generated and in a lower Self-BLEU score consequently.

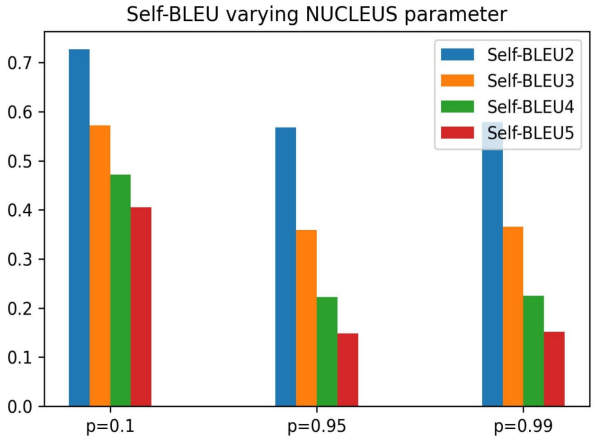


Figure 4

#### 5.4.4 Penalty

In previous experiments, generation has been done using a default value of penalty parameter equal to 1.2. If we modify penalty parameter value and set it to 1, generation will produce a output full of repetitions, but with a higher Self-BLEU.
If *penalty = 2*, instead, generation will produce captions avoiding repetitions, but with a lower Self-BLEU.

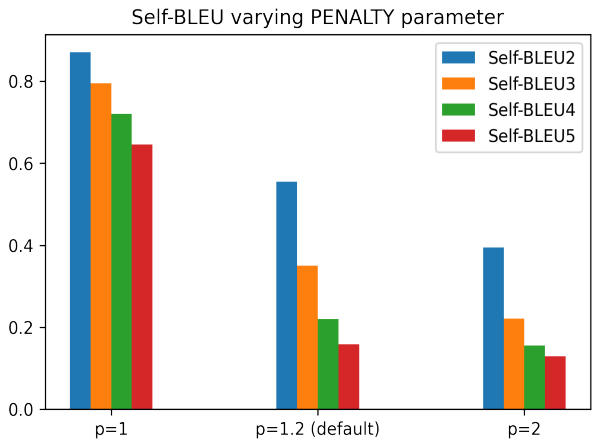


Figure 5

Human judges strongly preferred the texts generated with the anti-repetition methods, even though the corpus-based Self-BLEU score of such texts is lower with a higher penalty value. [5]

## 6. Future works

A possible improvement that could be done on our work is producing a text which is conditioned not only by a specific topic but also by an emotion or a context.
For this aim it could be useful to choose another appropriate dataset on which performing the training or integrate somehow COCO dataset, in order not to be too bounded to the "image's caption" concept.

## 7. Conclusions

During our work, we were able to generate conditional text based on a specific topic through an explicit control code. In this way, the generation can be easily controlled by humans.
After computing the metrics both in training and in test set, we discovered that the value obtained using the test set as reference were lower than the others, but still satisfying.
Furthermore, we did not find a significant difference between the models trained on different epochs or with different dataset size, while the most important difference, in terms of the quality of generated text, was found when the generation parameters (Temperature, Top-K, Nucleus, Penalty) changed, with a good trade-off using intermediate values.
In general, the captions generated were understandable by the human point of view, despite very few grammar mistakes that were not caught by BLEU, but only through a detailed observation of the generated captions.

## References


[1] Y. D. W. W. S. H. Y. S. Z. Y. Bin Guo, Hao Wang. Conditional text generation for harmonious human-machine interaction. 2020.

[2] https://cocodataset.org/download. Coco dataset.

[3] https://github.com/glample/fastBPE. fastbpe.

[4] T. W. W.-J. Kishore Papineni, Salim Roukos. Bleu: a method for automatic evaluation of machine translation. 2002.

[5] M. W. Mary Ellen Foster. Avoiding repetition in generated text. 2007.

[6] L. R. V. C. X. R. S. Nitish Shirish Keskar, Bryan McCann. Ctrl: A conditional transformer language model for controllable generation. 2019.

[7] P. von Platen. How to generate text: using different decoding methods for language generation with transformers.

[8] J. P. J. H. C. L. Yu Duan, Canwen Xu. Pre-train and plug-in: Flexible conditional text generation with variational auto-encoders.