Conditional text generation: leveraging slangs of the web

Luca Bajardi
Politecnico di Torino
s275782@studenti.polito.it

Ludovico Bessi
Politecnico di Torino
s275484@studenti.polito.it

Francesco Saracco
Politecnico di Torino
s270274@studenti.polito.it

Abstract

The main problem of conditional text generation is that it is mainly based on the content of an input set of examples: this leads to little diversification of the generated text. To overcome this shortcoming, we have fine tuned CTRL using three different datasets. The first model has been used as a baseline for comparison, while the other two have been used to obtain more formal and informal text. The BART model has been employed for text classification to gauge formality.

1. Introduction

In recent years, text generation has made great progress trying to become as flexible as the human language, which adapts its expressions depending on the context, topic or emotion. Thus, *conditional text generation* was defined. The model we have researched is *CTRL* [3], which can generate text conditioned on control codes that specify domain, style, topics, dates, entities, relationships between entities, plot points, and task-related behavior.

We initially fine tuned CTRL using 21K, 42K and 84K COCO captions from COCO [5] and compared the models using the BLEU [6], SELF-BLEU [9] and POS-BLEU [7] metrics. The main problem with this generation approach is that it is based on the content of an input set of examples, so we used two additional datasets: COCO captions mixed with *wikipedia* articles [2] to get more formal outputs and COCO captions mixed with *Reddit* comments [8] to get more informal outputs.

2. Model

Given a sequence $x = (x_1, \ldots, x_n)$, where each x_i is taken from a fixed set of symbols, the aim of language modeling is to learn the probability distribution $p(x)$ that produces the sequences. Being x a sequence, the probability distribution is naturally unfolded with the usual chain rule

$$\mathbb{P}(x) = \prod_{i=1}^{n} \mathbb{P}(x_i | x_{<i})$$

and thus language modeling is easily framed in the next-token prediction task.

CTRL is a conditional language model, meaning that the training phase is always conditioned on a control code c and the distribution learned is $p(x|c)$. The distribution can still be decomposed using the chain rule of probability that takes the control code into account:

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

This variation provides a form of control over the generation of new text, which will be produced accordingly to the code provided.

Data for the training is preprocessed and tokenized with fastBPE[1] as in [3]. Data is treated as a single stream of tokens. The stream is chunked into contiguous sequences of 256 tokens. Each sequence 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.

Pre-trained CTRL implementations are available as TensorFlow model, which is the original version by Salesforce[2], and as PyTorch model, in the context of Huggingface's Transformers repository[3]. We have used the former. You can find our scripts in our repository `https://github.com/luca-bajardi/Conditional_Text_Generation`

3. Fine tuning and experiments

First of all, we dedicated ourselves to producing a CTRL model successfully fine-tuned on the COCO captions dataset. The aim was to obtain a model able to complete the initial part of a test caption in a convincing way. A caveat is necessary before going into the details: all training phases have been carried out on 6 vCPUs of a Microsoft Azure virtual machine, instead of using the GPU. The reason for this choice is simply that we were not able, given

[1] https://github.com/glample/fastBPE
[2] https://github.com/salesforce/ctrl
[3] https://huggingface.co/transformers/model_doc/ctrl.html

the circumstances, to access GPUs with enough main memory to execute the training. Consequently, we were forced to be as frugal as possible in the implementation details and the dimension of the training datasets was somewhat limited. Still, we obtained pretty notable results. We also performed some comparisons with a smaller implementation of the model, but the latter had results that did not compete with those obtained via the original implementation, albeit trained on CPU.

3.1. Fine tuning of baseline model

With that said, we performed three subsequent phases of fine-tuning, having respectively 250, 250 and 500 iterations (250 iterations approximately correspond to 21k captions). The second and third phases were started on the checkpoint produced by the previous phase, so the models obtained had been trained respectively on 250, 500 and 1000 total iterations. The training data were simply obtained by prepending each COCO caption with the control code "caption" and using the result to build TensorFlow records. Each record contained 21 captions on average. All models obtained showed convincing results, so we picked the best one according to the metrics presented below in the paper, tested on 1000 test captions.

3.2. Fine tuning with influence on the writing style

As a second, more experimental step, we asked ourselves if it was possible, once having obtained a model able to complete captions, to influence the writing style of the captions produced. Specifically, we tried to complete the same set of test captions in a more formal and in a more informal style with respect to the baseline model. To this aim, we obtained two additional datasets of text, containing 5000 phrases each, respectively coming from Wikipedia and Reddit. Given that captions of images are usually short sentences, these phrases were selected to have length approximately equal to the average length of COCO captions, so to avoid to negatively influence the length of the output generated. To obtain the final result, two different approaches were tested.

3.2.1 Mixture input approach

On the first try, we built two datasets composed of a mix of COCO captions and extraneous text, with a 3:1 proportion. These datasets were then used to fine-tune the baseline model on two new control codes, "formal" and "informal". The idea for this process came from the concept of mixture probability distribution. As explained above, a language model learns a probability distribution corresponding to a corpus of text with the aim of producing new text from the same distribution. Intuitively, we may think that a particular writing style corresponds to a specific distribution on the vocabulary. So, COCO captions may be produced by the distribution $p_c(x)$, while texts from Wikipedia and Reddit are produced by two other probability distributions, respectively $p_w(x)$ and $p_r(x)$. In some sense, feeding the model with a mix of two different corpora of text forces it to learn the mixture distribution corresponding to the same relative weights. For example, through the training on a 3:1 mix of COCO captions and Wikipedia text, the model learns the distribution $p_{c,w}(x) = \pi \cdot p_c(x) + (1 - \pi) \cdot p_w(x)$ with $\pi = 0.75$. Obviously, all this reasoning does not pretend to be properly formal and has to be taken as intuition only.

3.2.2 Incremental learning approach

On the second try, we attempted to start from the previous training on COCO captions. The idea was to "bend" the language model learned on captions respectively towards a more formal and a more informal style, by means of the extraneous text. So, starting from the best baseline model, we executed 250 additional iterations on the "caption" control code, but this time using Wikipedia phrases as training input, to produce a more formal writing model. The same process was repeated for informal style, starting again from the baseline model. To the readers familiar with the concept, this second approach could somehow resemble the incremental learning framework. In this case, the previous knowledge consists of the ability of forming captions, while the incrementally learned part consists of the language style.

All the results of baseline fine-tuning and the described experiments are reported in the following section.

4. Results

4.1. Qualitative analysis of baseline model and generation strategies

Generation strategies play a major role in the effectiveness of LM models. Even state of the art models perform poorly when the process generating tokens is not carefully selected. There are many types of strategies, based on different ideas. For the quantitative evaluation part of our work, we decided to stick to the generation strategy conceived by the original authors of the model, for coherence and comparability with their results. This strategy is based on greedy sampling from the estimated probability distribution, with a repetition penalization mechanism. Before discussing quantitative analysis, we want to qualitatively compare different generation strategies on a small subset of captions, so to have a feeling about the effect of this feature in the generation process. In particular, we will observe the different results in completing the following prompts:

Four elephants with people riding
A white truck parked next

A group of people in
Several signs written in Arabic

The output presented below was generated by the best baseline model selected as explained in the quantitative analysis section. For greater insights on generation strategies (nucleus sampling, in particular) refer to [1].

4.1.1 Greedy sampling with repetition penalty

As stated above, this is the generation strategy chosen by the developing team of CTRL and is simply based on choosing the most probable token as the next word. On our selection of prompts, the results are as follows:

Four elephants with people riding them in a field.
A white truck parked next to a building.
A group of people in a field flying kites.
Several signs written in Arabic and English.

Actually, our tests showed that, in this particular application, the repetition penalty has no effect. The reason is obviously that we generate very short pieces of text and the model does not have enough space to repeat itself, but for longer generation sequences, this parameter becomes critical.

4.1.2 Temperature-based sampling

Another popular generation strategy, whose inspiration comes from statistical thermodynamics, is based on shaping the probability distribution dividing the estimated logits by a parameter t. If $0 < t < 1$, this procedure has the effect of magnifying the probability peaks, while $t > 1$ shapes the distribution towards a more uniform one. The next token is then picked sampling from the obtained distribution. In the following, we list the results with three different values of temperature.

Temperature: 0.5

Four elephants with people riding on their backs.
A white truck parked next to a fire hydrant.
A group of people in a field flying kites.
Several signs written in Arabic next to a building.

Temperature: 1

Four elephants with people riding atop top of their backs.
A white truck parked next to the fence.
A group of people in a kitchen sitting around food dating back to the 30's
Several signs written in Arabic and English, including one indicating safety.

Temperature: 1.1

Four elephants with people riding across high.
A white truck parked next to a house
A group of people in near rows of chairs with dogs
Several signs written in Arabic resident on them stand around an ornate building.

Temperature: 1.5

Four elephants with people riding dougoals wait impassively for Unicorn athlete saving grace to take bounding status westanna, wilstonmiddle asy, shadow shark.
A white truck parked next to the roadEnteretip holy wavewaveparade carrying peoplenefers flying fancy ...
A group of people in a grass field linkmode togetherpairs beautifully.
Several signs written in Arabic taps.

A value $t = 0.5$ seems to be a good trade-off between greedy sampling and sampling from the original distribution, which already shows some signs of degrading in the quality of generated text. In fact, in the latter case the text maintains grammatical correctness, but starts losing logical sense. Temperature values greater than 1 rapidly result in complete gibberish. This happens because flattening the distribution has the effect of elevating the tails of the distribution and lowering the peaks. Consequently, a much greater number of tokens ends having a non-negligible probability of being chosen, including useless ones. From this examples, it should be sufficiently evident that the selection of this parameter is not trivial and greatly influences the final result.

4.1.3 Top-k sampling

As the name suggests, this strategy relies on distribution truncation. At every generation step, the tokens with the top k highest probabilities are selected, while the rest of the vocabulary is discarded. The corresponding probabilities are rescaled to obtain a proper distribution to sample from. The following examples are obtained with various values for k and temperature equal to 1.

Top 5:

Four elephants with people riding them
A white truck parked next to a building.
A group of people in front of a large building.
Several signs written in Arabic on the roadside.

Top 15:

Four elephants with people riding one at the side of the field
A white truck parked next to a building.

A group of people in a restaurant sitting together.
Several signs written in Arabic are in front of a building.

Top 50:

Four elephants with people riding elephants as a line of people gather near.
A white truck parked next to a building.
A group of people in the city plaza eating food.
Several signs written in Arabic next to a building.

Again, the particular selection for the parameter k is critical for the final result. Different combinations with the temperature parameter could be tested, but this doubles the complexity of finding a suitable setting. Also, deeper research shows that the use of a constant value is sub-optimal. In fact, at every generation step, the next word distribution could be flat on a great number of reasonable words or peaked on few most probable words. Clearly, a fixed value cannot fit the particular shape of the distribution at every step, resulting in generic text in the first case and inappropriate text in the latter. A more advanced strategy should adjust dynamically the distribution truncation to its shape and this is what the next strategy does.

4.1.4 Nucleus sampling

In this last paragraph, we put on test a recent proposal for generation strategy. As anticipated above, it tries to exploit dynamically the shape of the distribution and, at the same time, to avoid the inconvenient of having to choose the parameters' values. In fact, at every step the shape of the distribution is used to guide the truncation phase. Given the distribution $\mathbb{P}(x|x_{<i})$ at step i, the algorithm finds the smallest set $V^{(p)}$ such that

$$\sum_{x \in V^{(p)}} \mathbb{P}(x|x_{<i}) > p$$

for a fixed threshold p. The token are selected and the corresponding probabilities are rescaled to obtain a proper probability distribution on $V^{(p)}$, from which the next word is sampled. Higher values for p result in smaller sets of words. With respect to the previous method, this procedure consists in a dynamic setting of k depending on the shape of the distribution, thus overcoming the major limitation of a fixed value. Although also in this case a parameter selection is needed, p is a much more intuitive and comprehensible quantity and its selection is easier. As showed below, diverse settings of this parameter all result in good quality generation.

$p = 0.95$

Four elephants with people riding elephants out in a field.
A white truck parked next to some motorcycles.
A group of people in a large tent play it.
Several signs written in Arabic in the town of Nimrn.

$p = 0.80$

Four elephants with people riding in their sides.
A white truck parked next to a line of wooden benches.
A group of people in a restaurant sitting around at a table.
Several signs written in Arabic near a stop sign.

$p = 0.50$

Four elephants with people riding them are in a field.
A white truck parked next to a fire hydrant.
A group of people in a room with various computers.
Several signs written in Arabic on a wall.

The examples show that, among the analyzed methods, nucleus sampling is the only one that manages to compete with the default generation strategy of CTRL model. On bigger test sets, nucleus sampling proves to generate a slightly more varied text with respect to the latter method.

In conclusion, we showed that the choice of the generation strategy is not trivial and is as critical as other, more architectural aspects in a language modelling task. The results obtained vary a lot from a strategy to another.

4.2. Quantitative analysis

We have used three different metrics to evaluate the quality of text: BLEU [6], SELF-BLEU [9] and POS-BLEU [7].

BLEU (bilingual evaluation understudy) uses a modified form of *precision* to compare a candidate sentence against multiple reference texts.

The modification that BLEU makes is fairly straightforward. For each word in the candidate sentence, the algorithm takes its maximum total count, m_{max}, in any of the references.

For the candidate sentence, the count m_w of each word is clipped to a maximum of m_{max} for that word. These clipped counts m_w are then summed over all distinct words in the candidate. This sum is then divided by the total number of unigrams in the candidate sentence.

In practice, however, using individual words as the unit of comparison is not optimal. Instead, BLEU computes the same modified precision metric using n-grams. The length which has the "highest correlation with monolingual human judgements" was found to be four [6]. The unigram scores are found to account for the adequacy of the text, that is how much formation is retained. The longer n-gram scores account for the fluency of the text.

SELF-BLEU was introduced to evaluate just variety of sentences. It measures BLEU score for each generated sentence by considering other generated sentences as reference.

By averaging these BLEU scores we obtain SELF-BLEU: lower values shows more diversity in the generated text.

POS-BLEU is BLEU score calculated on POS (part of speech) tags instead of words.

On top of that, in order to evaluate our own defined control codes, we have used BART [4] model from transformers library to classify how much a given string is formal.

4.2.1 Baseline model

First of all, we present the results obtained by the models trained on COCO captions with an increasing number of iterations.

Metric	250 iters	500 iters	1000 iters
BLEU2	0.534	0.541	0.539
BLEU3	0.499	0.507	0.505
BLEU4	0.462	0.470	0.467
BLEU5	0.409	0.419	0.416

Metric	250 iters	500 iters	1000 iters
S-BLEU2	0.773	0.766	0.763
S-BLEU3	0.604	0.592	0.587
S-BLEU4	0.421	0.425	0.419
S-BLEU5	0.324	0.306	0.309

Metric	250 iters	500 iters	1000 iters
P-BLEU2	0.684	0.687	0.685
P-BLEU3	0.624	0.629	0.627
P-BLEU4	0.571	0.578	0.574
P-BLEU5	0.512	0.520	0.516

It is also interesting to see how many sentences changed from one model to the other.

193 sentences from the model with 250 iterations are identical to the model with 500 iterations. Let's now calculate the BLEU and POS-BLEU score taking as reference the model with 250 iters and as output the model with 500 iters:

Metric	250 vs 500 iters
BLEU2	0.693
BLEU3	0.665
BLEU4	0.637
BLEU5	0.602

Metric	250 vs 500 iters
P-BLEU2	0.793
P-BLEU3	0.769
P-BLEU4	0.724
P-BLEU5	0.685

168 sentences from the model with 250 iterations are identical to the model with 1000 iterations. Let's now calculate the BLEU and POS-BLEU score taking as reference the model with 250 iters and as output the model with 1000 iters:

Metric	250 vs 1000 iters
BLEU2	0.683
BLEU3	0.653
BLEU4	0.623
BLEU5	0.584

Metric	250 vs 1000 iters
P-BLEU2	0.796
P-BLEU3	0.755
P-BLEU4	0.719
P-BLEU5	0.681

4.2.2 Formal and informal models

We now analyze how much the formality level of the sentences changed when when trying to influence the style of generated captions. Letting F_i be the formality of a sentence and N the number of sentences, we define the average formality level of the output as:

$$AFL = \frac{\sum_i F_i}{N}$$

Likewise for informality level:

$$AIL = \frac{\sum_i I_i}{N}$$

As mentioned, we tried two different approaches. In both of them, we used the best performing model according to BLEU-4, that is the model with 500 iterations. First, we found the AFL and AIL for the starting model: $AFL = 0.2484$ and $AIV = 0.7515$.

4.2.3 Mixture input approach

We recall that, in this first approach, we started from the trained model on the captions and added two different control codes: "formal" and "informal". They are trained on a mix of caption and new style 3:1.

Metric	baseline	wikipedia	reddit
BLEU-4	0.470	0.464	0.467
AFL	0.258	0.248	0.249
AIL	0.751	0.752	0.751

These quantitative metrics turn out to be not very informative, some qualitative examples are given below.

Outputs that have the maximum difference in informal value:

Normal sentence: *'A living room and dining area with a large fireplace.'*
Informal sentence: *'A living room and dining area with a large screen tv on the wall.'*

Outputs that have the maximum difference in formal value:

Normal sentence: *'A man in a white shirt and black shorts is holding his tennis racket.'*
Formal sentence: *'A man in a white shirt and hat is holding an umbrella.'*

4.2.4 Incremental learning approach

Metric	baseline	wikipedia	reddit
BLEU-4	0.470	0.422	0.436
AFL	0.249	0.250	0.326
AIL	0.751	0.750	0.674

These quantitative metrics turn out to be not very informative, some qualitative examples are given below.

Outputs that have the maximum difference in informal value

Normal sentence: *'A group of people at a table with plates and glasses.'*
Informal sentence: *'A group of people at a table having fun eating pizza.'*

Outputs that have the maximum difference in formal value:

Normal sentence: *'A baby boy wearing a hat and holding an umbrella.'*
Formal sentence: *'A baby boy wearing a striped shirt and bow tie.'*

4.2.5 Comments on quantitative results

From the plain numbers, the influence on the style of the generated captions does not seem to be very evident. However, this fact is also due to the model used to evaluate the formality level, which obviously is not able to catch all the subtleties of human language. From a qualitative inspection, the difference appears much more evident. To prove this, we present below some examples from the two models trained with additional iterations on control code "caption" using extraneous text, whose outputs show the most pronounced effects.

Formal: *A group of people are gathered together to listen and learn about the latest developments in science or technology.*
Informal: *A group of people are standing around a table having fun posing for the camera*

Formal: *A bunch of different people are standing around a table with plates and glasses.*
Informal: *A bunch of different people have this name so it must be a cool name*

Formal: *An old, black and white photo of a man riding on top of an elephant.*
Informal: *An old, black and white photo of a man that looks like Moe from The Simpsons*

Formal: *A group of horses gather together to drink from a watering hole in the middle of the field.*
Informal: *A group of horses gather around a watering hole in the middle of nowhere*

Formal: *A tabby cat is shown with its eyes wide open.*
Informal: *A tabby cat is shown wearing a tie.*

A possible reason for the scarce sensitivity of BERT model to the formality/informality level of the generated captions is that the difference in formality between many pairs of captions generated lies more in the content than in the style of language used and BERT is not able to detect this aspect. This is possibly due to the choice of the text used to influence the style, particularly Wikipedia, that presents a higher level of technical language rather than a higher level of formal language with respect to Reddit text. Despite all these considerations, from a human point of view the difference between the language style of the two models appears very evident and and we consider the overall result to be successful.

5. Future directions

There are different ways this work could be further improved. Fine tuning the text classification model on Reddit and Wikipedia sentences could greatly improve its accuracy, providing sounder insights to our results. Also, due to time constraints, we were not able to make a careful research and experimentation of the extraneous texts to use in the style-influencing part. As also considered above, a better selection of this important experimental factor could lead to better results. This also involves the choice of the mix proportions: different relative weights could be tested.

References

[1] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi. The curious case of neural text degeneration, 2020.

[2] Kaggle. Wikipedia dump august 2018. 2018.

[3] N. S. Keskar, B. McCann, L. R. Varshney, C. Xiong, and R. Socher. CTRL: A conditional transformer language model for controllable generation. *CoRR*, abs/1909.05858, 2019.

[4] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *Facebook Research*, 2019.

[5] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[6] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. *ACL-2002*, 2002.

[7] M. Popovic and H. Ney. Syntax-oriented evaluation measures for machine translation output. *ACL*, 2009.

[8] P. shift API. Reddit dump march 2008. 2008.

[9] Y. Zhu, S. Lu, L. Zheng, J. Guo, W. Zhang, J. Wang, and Y. Yu. Texygen: A benchmarking platform for text generation models. *SIGIR*, 2018.