Conditional Text Generation with CTRL

Giulio Cerruto
Politecnico di Torino
277335@studenti.polito.it

Lorenzo Paletto
Politecnico di Torino
s278780@studenti.polito.it

Marco Saponara
Politecnico di Torino
s277323@studenti.polito.it

Abstract

In this report we approach the task of conditional text generation through the use of the Conditional Transformer Language Model, CTRL. After fine-tuning the pretrained model on the COCO dataset for image captioning, we generate several meaningful captions. We set the results obtained with the default 48-layer model as baseline, then we experiment by changing the parameters of the 36-layer model and make multiple generations. Finally, we asses the quality of our results using ad-hoc metrics (BLEU, Self-BLEU, POS-BLEU, METEOR), in order to understand the impact of each hyperparameter and to determine which is the most suitable configuration for this case study.

Figure 1: image 318556

1. Introduction

In the last years, text generation has made remarkable progress since the first models relying on rules and statistical inference. Indeed, Deep Learning allows us to overcome the limitations due to modeling the complexity and the variety of human language with a set of fixed rules and statistics. Nowadays, text generation is performed by several architectures mainly relying on neural networks, which can use input data of various types and produce sentences of excellent quality.

This task can be divided in two groups:

- *Unconditional* text generation, based on the content of an input set of examples provided during the training phase;
- *Conditional* text generation, which is influenced also by external conditions, such as the context, the topic or the emotion.

For this reason, the latter is suitable for more complex NLP problems, such as goal-oriented conversations, Query & Answering (Q&A) systems and image captioning, that is the particular framework of this project.

More precisely, our goal is to generate captions by conditioning the text on the content of specific images. In order to achieve it, we exploit CTRL ([3]), a conditional transformer language model that will be further explored in the next sections.

The object of the present work is the evaluation and the analysis of the performances of the model under several different configurations of its hyperparameters, involved both in the training and generation phases, to better understand which are the most relevant ones. The scripts, as well as other useful information, are available at the following Github repository.

2. Dataset

All the experiments have been performed on the Common Objects in Context (COCO) dataset ([4]), usually exploited for large-scale object detection, segmentation and image captioning. It is freely available at the following link. Since we are interested in the captioning task, let us focus on data from 2014. They are divided in three sets:

- a training set, *captions_train2014.json*;
- a validation set, *captions_val2014.json*;
- a test set, *image_info_test2014.json*.

Multiple different captions are available for each caption: more precisely, we have an average of five captions per image inside the training set, in fact it contains 82783 images and 414113 captions.

As an example, we show the captions of image 318556 (fig. 1):

> A very clean and well decorated empty bathroom.
> A blue and white bathroom with butterfly themed wall tiles.
> A bathroom with a border of butterflies and blue paint on the walls above it.
> An angled view of a beautifully decorated bathroom.
> A clock that blends in with the wall hangs in a bathroom.

The test set, instead, does not have captions associated to images but it provides a list of *names* and *supercategories* related to the possible content of an image. As we can see from Table 1, there are 12 supercategories and 80 names related to them.

3. Model

Our work fully relies on CTRL model by *Salesforce*, a conditional transformer language model able to generate text conditioned on control codes that specify domain, style, topics, dates, entities, relationships between entities, plot points and task-related behavior.

supercategory	names
accessory	*backpack, umbrella, handbag, tie, suitcase*
animal	*bird, cat, dog, horse, sheep, cow, elephant, bear, zebra, giraffe*
appliance	*microwave, oven, toaster, sink, refrigerator*
electronic	*tv, laptop, mouse, remote, keyboard, cell phone*
food	*banana, apple, sandwich, orange, broccoli, carrot, hot dog, pizza, donut, cake*
furniture	*chair, couch, potted plant, bed, dining table, toilet*
indoor	*book, clock, vase, scissors, teddy bear, hair drier, toothbrush*
kitchen	*bottle, wine glass, cup, fork, knife, spoon, bowl*
outdoor	*traffic light, fire hydrant, stop sign, parking meter, bench*
person	*person*
sports	*frisbee, skis, snowboard, sports ball, kite, baseball bat, baseball glove, skateboard, surfboard, tennis racket*
vehicle	*bicycle, car, motorcycle, airplane, bus, train, truck, boat*

Table 1: List of categories and supercategories

3.1. Language modeling

Usually, the language modeling task is achieved by treating a sentence as a sequence of words, $x = (x_1, \ldots, x_n)$ where x_i belongs to a fixed vocabulary, and factorizing its probability distribution, $p(x)$, using the chain rule:

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

This decomposes language modeling into next-word prediction.

Similarly, CTRL provides a language model conditioned also on a control code c and, by consequence, aims at learning the distribution $p(x|c)$, that can be still decomposed as

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

by training over a large corpus of textual data, $D = \{x^1, \ldots x^{|D|}\}$, prepended with a specific control code, c^*, and minimizing the loss

$$\mathcal{L}(D) = -\sum_{k=1}^{|D|} \log p_\theta(x_i^k|x_{<i}^k, c^*).$$

3.2. Architecture

In practice, CTRL acts as a standard Transformer ([7]). An input sequence containing n tokens is embedded as a sequence of n corresponding vectors in $\mathbb{R}^d$, where $d = 1280$. Each vector is the sum of a learned token embedding and a sinusoidal positional embedding. This sequence of vectors is stacked into a matrix $X_0 \in \mathbb{R}^{n \times d}$, so that it can be processed by l attention layers, $l \in \{36, 48\}$. The i-th layer consists of two blocks, each of which preserves the model dimension d. The core of the first block is multi-head attention with $k = 16$ heads that uses a causal mask to preclude attending to future tokens:

$$\begin{aligned}
&\text{attention}(X, Y, Z) = \text{softmax}\left(\frac{\text{mask}(XY^T)}{\sqrt{d}}\right) Z,\\
&\text{multiHead}(X, k) = [h_1, \ldots h_k]W_o,\\
&\text{where} \quad h_j = \text{attention}(XW_j^1, XW_j^2, XW_j^3).
\end{aligned}$$

The core of the second block is a feedforward network with ReLU activation that projects inputs to an inner dimension $f = 8192$, with parameters $U \in \mathbb{R}^{d \times f}$ and $V \in \mathbb{R}^{f \times d}$:

$$FF(X) = \max(0, XU)V.$$

Each block precedes core functionality with layer normalization and follows it with a residual connection. Scores for

each token in the vocabulary are computed from the output of the last layer:

$$\text{scores}(X_0) = \text{layerNorm}(X_l)W_{\text{vocab}}.$$

During training, these scores are the inputs of a cross-entropy loss function. During generation, the scores corresponding to the final token are normalized with a softmax, yielding a distribution for sampling a new token.

Given a control code and a prompt, the model is able to generate the next words. At each iteration the *top k* most suitable words are identified and the following probabilities are assigned:

$$p_i = \frac{\exp(x_i/T \cdot I(i \in g))}{\sum_j \exp(x_j/T \cdot I(j \in g))}.$$

where I is a discounting term decreasing the probability of a word i to be generated if it belongs to the g previously generated ones and T is a constant called *temperature*. If $T \to 0$ it approximates a greedy distribution which magnifies the peaks in the probability distribution, if $T \to \infty$ it flattens the distribution to make it more uniform. The next token is then chosen by sampling through a multinomial distribution.

4. Method

4.1. Data Preprocessing

Since our analysis does not involve images, the main preprocessing procedure of our pipeline consists in the assignment of a set of categories to each caption in the dataset, as no clear connection is provided between the two. Therefore, we decide to assign a specific category to a caption only when the former (or the associated supercategory) is contained inside the caption's sentence.

In order to achieve this task, we perform a research of the 13 supercategories by looking for the string in each caption and, in case one is found, we assign all the categories associated to that specific supercategory. If no supercategory is found we increase the precision of our research by searching the single categories inside each caption. We decide to give particular attention to the supercategory *person* because it includes only the homonymous category, not only looking for the word *person* but also for other words, e.g. *man/men*, *woman/women*, as well as *people*, which are equally valid.

Note that the list of categories provided by the test set is not exhaustive: in fact, it is possible to increase the number of categories to 91, by adding the names *street sign*, *hat*, *shoe*, *eye glasses*, *plate*, *mirror*, *window*, *desk*, *door*, *blender*, *hair brush*.

With this strategy we manage to assign categories to approximately 82% of the entire dataset. The remaining unlabeled captions are assigned to an ad-hoc category called *general*.

4.2. Fine-Tuning on Custom Dataset

Given a sample of textual data sharing the same control code, it is possible to fine-tune the model on it, in order to augment existing control codes or add new ones. Let us focus on the latter case.

Since each of the 91 categories is now associated to a specific set of captions, it is natural to condition the language model on them.

The training step, that is described in detail at the following link, mainly consists in two parts:

1. conversion of the raw text input data into TFRecords;
2. fine-tuning the model on these TFRecords files.

The only modification we did to the original training script provided by *Salesforce* involves the possibility to set the learning rate of the optimizer.

Because of computational issues related to the size of the model and the computational power available, no big dataset can be input to the fit method. To overcome this issue, only captions labelled with *person* are used for the training. This choice is legitimated by the size of the dataset obtained after this filtration (164688 captions) and by the heterogeneity of the words used in this context: since they involve people in different settings, the extent of the vocabulary is quite large (on the contrary, more specific categories such as *electronic* would have given quite less diversification in the sentences). Furthermore, to deal with the large size of the model, a procedure of *manual batching* needs to be exploited: 40000 captions partitioned into four equal sized datasets are fed to the training phase one after the other, rather than all at once.

The training procedure strongly depends on the quality of the input data, e.g. the coherency between captions and the number of available samples, but also on the number of training epochs. Note that the model will likely overfit when trained on a limited amount of data and will end up memorizing the input captions. In this case it is suggested to set a low number of iterations, preferably equivalent to 1-10 epochs.

4.3. Text Generation and Evaluation

After having trained the model on the new control codes, its weights are saved inside a checkpoint. This is then reached by the generation script, where a new text is created starting from a prompt that necessarily contains a control code, but can also contain other control codes or the first words of the sentences to be generated. The generation phase relies on 1000 prompts, obtained by sampling from the original dataset and truncating the captions at the second word.

To conclude, the quality of the output is evaluated through four natural language processing metrics: BLEU,

Self-BLEU, POS-BLEU and METEOR (described in more detail in Section 6). Another 1000 captions extracted from the original dataset are used as a reference for the evaluation of the metrics on the generated sentences.

5. Experiments

In this section, we analyze the meaning and the purpose of the main hyper-parameters of the model and the impact that changing their value has on the generated caption. The hyper-parameters taken into account are: *top k*, *temperature*, the *learning-rate* of the optimizer, the number of *iterations* and the *top-p* nucleus sampling.

5.1. Top-k

The *top-k* parameter controls the number of tokens on which the model performs a random sampling. Every time a new token is to be generated, the *k* tokens with the highest probabilities are selected. From this set, one token is randomly sampled and chosen as the next word. This is useful to avoid generating the same sentence when starting from the same prompt multiple times. Increasing the value of *top-k* results in a larger variety of generated sentences but might, in turn, reduce the quality of the results. Here we show some captions obtained varying the value of the parameter.

top 4

A man is standing next to an umbrella and this is a great place to get the most out of your time.
A man is standing next to the bus stop.
A man is walking on a street with an umbrella and a dog.

top 8

A man sitting in front of an umbrella.
A man sitting on the ground near a group of people.
A man in a yellow jacket and hat sits next to the bus stop.

top 10

A man is standing next to the bus stop.
A man in a yellow jacket and hat sits next to the girl.
A man is walking on a street with an umbrella and a dog.

top 15

A man sitting in front of an umbrella.
A man is standing next to the bus stop.
A man in a suit sits next to an old white bus.

5.2. Temperature

The *temperature* parameter, T, is introduced in order to avoid the choice of the next token to be deterministic: the higher the *temperature*, the flatter the probability distribution on the next *top k* possible tokens; on the contrary, the lower the *temperature*, the sharper the peak of the distribution. In the former case, the choice of the parameter *top k* plays a much more important role than in the former. Let us report some captions obtained varying the value of the parameter.

temperature 0.2

A woman is sitting on a park bench
A man sits next to a group of people at a bus stop.
Woman standing next to a man holding an umbrella

temperature 0.4

A woman is sitting on a park bench.
A man sits next to a girl in the grass at an intersection
Woman standing next to a person holding an umbrella

temperature 0.6

A woman stands next to a man on top of a bus
A man sitting on the sidewalk at an intersection holding his hand out
Woman standing next to a bus stop.

temperature 0.8

A woman sits on a bench
A man is sitting on the ground next to an empty red and blue bus.
Woman standing next to a fire extinguisher.

5.3. Learning-rate

Learning rate determines the step size at each iteration while moving toward a minimum of the loss function. Practically, this parameter can be thought of as how fast the model is learning. A low value can lead to the algorithm not being effective enough at acquiring new information, while a high value may result in the loss function 'jumping over' the minimum and therefore missing it. Here we show some captions obtained varying the value of the parameter.

learning rate 0.001

A group of people
A man and a woman
People on the street

learning rate 0.01

A group of people sitting on the street
A man sits next to a girl in the grass at an intersection
Woman standing next to a person holding an umbrella

learning rate 0.1

A group of people
A woman and a man
A street scene of a group of people walking by the beach

5.4. Iterations

The number of *iterations*, or *epochs*, defines the number of times that the learning algorithm will work through the entire training dataset. A low number of epochs may prevent the model from learning enough to produce quality sentences, while a high number of iterations may result in the captions being learnt by heart from the algorithm. Let us report some captions obtained varying the value of the parameter.

2 iterations

A man
A group
Several people

5 iterations

A room full of people sitting on a park bench.
A woman is sitting on a park bench.
A man in a yellow jacket and hat sits next to the girl

10 iterations

A woman is sitting on a bench in front of an old man.
The man looks at his phone and then turns it back
A woman is sitting on the back of an elephant

5.5. Nucleus

Top-p nucleus sampling is an alternative approach to *top-k sampling*. Instead of sampling only from the most likely k words, this approach chooses from the smallest possible set of words whose cumulative probability exceeds the probability p. The probability mass is then redistributed among this set of words. This way, the number of words in the set can dynamically increase and decrease according to the next word's probability distribution [5]. Increasing the value of p results in a larger variety of generated sentences, but might, in turn, reduce the quality of the results. Here we show some captions obtained varying the value of the parameter.

nucleus 0

A man is walking down the sidewalk with his dog next
A couple of people are sitting on a park bench
A man is sitting in the middle of an intersection

nucleus 0.2

A view of a man in the middle of an intersection
A woman is walking down the street
A woman is walking down the street with a dog in her arms.

nucleus 0.5

A view of a group of people walking down the street
A man is standing on top of an old wooden
A woman is walking down the street with a dog in her arms.

nucleus 0.8

A woman is walking down the street with a dog on her back
Woman standing on a street corner in front of an apartment building.
A man is walking down the sidewalk with his dog

6. Evaluation

6.1. Metrics

In order to evaluate our generated captions we test them with four Untrained Automatic Evaluation Metrics: BLEU, Self-BLEU, POS-BLEU and METEOR. These metrics compute a score that indicates the similarity between an automatically generated text and human-written reference text. They are efficient, fast to compute and widely used in the text generation framework. We use the code functions provided by the NLTK library in Python.

All the metrics exploited are n-gram Overlap Metrics, which means they count the number n-words blocks that are present both in the reference and in the generated text. In this section we present them and briefly explain how they work and how they are computed.

6.1.1 BLEU

The BiLingual Evaluation Understudy (BLEU) is one of the first metrics used to measure the similarity between sen-

tences, it is based on the idea that the closer the generated text is to the reference, the better it is. It is calculated for every candidate sentence against all the reference ones, if there are more candidate sentences the mean of all the BLEUs gives the final result. It is computed as

$$\text{BLEU} = \min(1, e^{1-\frac{r}{c}})\Big(\prod_{i=1}^{4} \text{precision}_i\Big)^{1/4}$$

where:

- r = count of words in a reference text
- c = count of words in a candidate text
- precision_i is the precision calculated as

$$\frac{\text{n. of generated n-grams occurring in ref. sentence}}{\text{total n. of n-grams present in generated sentence}}$$

for every n-gram of length $i = 1, \ldots, 4$.

The formula consists of two parts: the brevity penalty and the n-gram overlap. The brevity penalty penalizes generated translations that are too short compared to the closest reference length with an exponential decay. The brevity penalty compensates for the fact that the BLEU score has no recall term. The n-gram overlap counts how many unigrams, bigrams, trigrams, and four-grams match their n-gram counterpart in the reference translations. This term acts as a precision metric. Unigrams account for adequacy while longer n-grams account for fluency of the translation.

BLEU is a good metric but comes with some issues, in fact literature ([2]) suggests it lacks of semantic coherence and it does not evaluate if the text has poor information content. It only considers precision to give a score and this is computed as "1 vs n" sentences, which might not be the best way.

6.1.2 Self-BLEU

This metric ([8]) is defined within the set of generated sentences. For every candidate sentence the standard BLEU metric is computed against all the other ones and the mean of all the scores is returned. This metric is useful to measure how much the sentences the model produces are similar or different to one another: a high value could mean too much repetitiveness, a low one too much randomness.

6.1.3 POS-BLEU

The third metric ([6]) is analogous to BLEU but is computed on POS (Parts Of Speech) rather than on n-grams. Every sentence is split in words and every word is converted in a POS-tagger, a label that identifies the word (nouns are labeled as "NN", adjectives as "JJ", etc).

6.1.4 METEOR

The Metric for Evaluation of Translation with Explicit ORdering [1] is a metric designed to address some of the issues found in BLEU and has been widely used for evaluating text generation models. It creates alignments between unigrams in the reference and generated text, also considering synonyms and words stemming.

It is computed as

$$\text{METEOR} = (1 - \text{penalty})\frac{10RP}{P + 9R}$$

where:

- P is precision computed as before but only for unigrams
- R is recall computed as

$$\frac{\text{n. of generated unigrams occurring in ref. sentence}}{\text{total n. of unigrams present in reference sentence}}$$

- $penalty$ is a penalty computed for the alignment considering mismatching of longer n-grams.

As for BLEU, the formula consists of two parts: penalty and n-gram overlap. This time the second part is calculated as an harmonic mean of precision and recall, where recall has more weight than precision.

METEOR is computed as "1 vs 1" sentence, if there are more reference sentences only the best score among all the references is returned and if there are more generated sentences the mean of all the scores is computed.

6.2. Results

In this section we show the results in terms of scores of the metrics. We let some of the model's hyper-parameters vary and report the obtained scores.

Let us start with our benchmark results, the 48 layer model trained with the recommended configuration.

Metric	48 layers
BLEU	0.482
Self-BLEU	0.838
POS-BLEU	0.141
METEOR	0.560

We now use the 36 layer model and vary its parameters one at the time. The first results come from the default configuration with $\textit{topk} = 10, \textit{temperature} = 0.4, \textit{epochs} = 5, \textit{learning rate} = 0.01$ and $\textit{nucleus} = 0$.

The results are very close to the ones we get with the more complex model but this time Self-BLEU is slightly worse, which implies more heterogeneous generated captions.

Metric	36 layers
BLEU	0.492
Self-BLEU	0.753
POS-BLEU	0.151
METEOR	0.553

The first parameter we vary is the topk, from 10 to 4, 8 and 15. By doing this we first decrease and then increase the number k of words from which the generated words are chosen at each step.

Metric	top-k=4	8	15
BLEU	0.490	0.489	0.486
Self-BLEU	0.808	0.783	0.781
POS-BLEU	0.152	0.152	0.152
METEOR	0.546	0.544	0.544

With top-k equal to 4 we see a slight increase in the BLEU score but also in the Self-BLEU which means that the fewer words the model can choose from when writing a new sentence, the more they are a duplicate of the reference captions.

Then we change temperature. By increasing and decreasing it we respectively flatten the probability distribution for the next word and sharpen it. This implies that in the first case all the possible next words tend to be equally probable while in the second one the more probable ones will tend to be even more probable.

Metric	T = 0.2	0.6	0.8
BLEU	0.553	0.413	0.343
Self-BLEU	0.857	0.718	0.633
POS-BLEU	0.153	0.139	0.130
METEOR	0.564	0.515	0.483

By comparing this table with the previous one we conclude that an increase in temperature corresponds to a decrease in the scores of the metrics: this means that the overall quality of the captions tends to be lower, but this is also due to the choice of the next word being wider. The set of scores obtained with temperature = 0.2 is higher than the one obtained with the recommended configuration.

We now change the number of training epochs from the recommended value of 5 to 2 and 10.

Metric	epochs = 2	10
BLEU	-	0.375
Self-BLEU	-	0.716
POS-BLEU	-	0.132
METEOR	-	0.463

With 10 epochs we do not see particular improvements while when the epochs are only 2 the model does not learn and prints only the prompts without continuing the sentences.

The next parameter is the learning rate which we change from 0.01 to 0.1 and 0.001.

Metric	l.r. = 0.1	0.001
BLEU	0.266	0.203
Self-BLEU	0.604	0.509
POS-BLEU	0.132	0.126
METEOR	0.361	0.345

The new configurations decrease the scores of the metrics, both in term of n-grams and in term of POS. Even if the first column has higher scores than the second we conclude that the model works better with the recommended value of 0.01.

The last parameter that we change is the nucleus, the probability *p* used in the *top-p* nucleus sampling. The recommended value is 0 and we try to increase it going closer to 1.

Metric	nucleus = 0.2	0.5	0.8
BLEU	0.448	0.457	0.447
Self-BLEU	0.925	0.941	0.842
POS-BLEU	0.145	0.144	0.136
METEOR	0.541	0.543	0.521

All the metrics that compare the generated text with the reference one get worse scores than the ones got with the standard configuration. The only metric that increases is Self-BLEU but only for the values 0.2 and 0.5: this means that the generated sentences get more heterogeneous with respect to the reference ones but at the same time they get more similar to each other.

7. Conclusions

Given the results of the previous section we conclude that the best performing configuration is the recommended one (*topk*=10, *temperature*=0.4, *epochs*=5, *learning rate*=0.01 and *nucleus*=0) over the 36-layer model. We base our decision on the trade-off between the BLEU, Pos-BLEU and METEOR scores and the Self-BLEU scores. The first group of metrics evaluates the similarity between generated and reference text while Self-BLEU penalizes repetitiveness. The only noteworthy parameter is temperature because all its variations produce different but meaningful results.

Despite being both heavier and longer to train, the 48-layers model does not perform better than the 36-layers in terms of metrics. The generated captions produce good scores in every metrics but get too high Self-BLEU values. Scores higher than 0.75 usually mean that the model tends to produce sentences too similar to one another. The reason

for this result could be either a too small number of captions used to train the model or a scarce number of training iterations.

Average scores from BLEU suggest that the model learns well and does not tend to copy entirely already existing captions. The vast majority of the captions has a complete sense and does not show lexical or syntactic mistakes. Even if the model is pre-trained every generated caption does not show traces of the previous training an is fully coherent with the COCO captioning contest.

8. Future works

Our work could be improved in several ways, for example by increasing the computational power and training the model on a larger dataset and running more iterations. This would lead to a more heterogeneous set of generated captions and would likely decrease the Self-BLEU score.

Another interesting improvement would be training the model on sets of captions associated to different labels. This could allow us to combine multiple control codes and consequently generate even more variegated captions, since COCO's images usually contain more than one entity.

One last possible (although very challenging) work could be combining this model with a neural network focused on the object detection task (eg. CNN). The output labels of the first network could be fed to the CTRL model as control codes in order to automatically generate a caption for the image.

References

[1] S. Banerjee and A. Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. 6

[2] A. Celikyilmaz, E. Clark, and J. Gao. Evaluation of text generation: A survey, 2021. 6

[3] N. S. Keskar, B. McCann, L. Varshney, C. Xiong, and R. Socher. CTRL - A Conditional Transformer Language Model for Controllable Generation. *arXiv preprint arXiv:1909.05858*, 2019. 1

[4] T. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. Microsoft coco: Common objects in context. *arXiv:1405.0312*, 2015. 1

[5] B. Mann. How to sample from language models. *towardsdatascience.com*, 2019. 5

[6] M. Popovic. Syntax-oriented evaluation measures for machine translation output. pages 29–32, 03 2009. 6

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2017. 2

[8] Y. Zhu, S. Lu, L. Zheng, J. Guo, W. Zhang, J. Wang, and Y. Yu. Texygen: A benchmarking platform for text generation models, 2018. 6