3.11.2023

Common Sense Generation

Gilat Toker

Roi Reichart & Nitay Calderon

1. Motivation and background

Natural Language Processing (NLP) algorithms are constantly improving and reaching significant milestones. However, such algorithms rely on the availability of sufficient data. Unfortunately, this assumption does not hold in many cases due to the labor-intensive and expensive nature of the data collection and preparation process. A natural alternative to costly human annotation would be to automatically generate new examples for model training. Doing so may expose the model to additional training examples and better represent the data distribution.

One way to generate new examples is to use a Commonsense Generation model, which is capable of generating human-like coherent texts from a given set of constraints. Typically, these constraints require the model's output to include specific words and terms. For example, given the set of words: {apple, pick, eat}, the commonsense generation model should generate a text that contains these words, such as, "A kid picked an apple from the tree and ate it".

2. Detailed problem definition

This project will focus on commonsense generation when the constraints on the model consist of a list of words, as outlined in the motivation. By performing this task, many different models can be benefited and improved. An example would be to take an existing training set and modify it to meet specific needs (for example, add sentences that address underrepresented populations in the data). Through this approach, the model will be trained on a more balanced set of training data and thus produce results that are more representative of the actual distribution of the world.

The model will be trained and tested using the Common_gen dataset. The dataset was constructed by combining crowdsourcing from AMT with existing caption corpora. Overall, it contains 30k concept sets and 50k sentences, which are arranged in pairs, so that each constraint is accompanied by several sentences.

For evaluating the quality of the results of the model, we consider two criteria: 1) whether the model meets all of the constraints (contains all the words in the sentence) and 2) whether the sentence is written in a manner that is consistent with human-like coherent language. To ensure that the first part is accomplished, an automatic test will be conducted to identify words based on their roots. To measure the second part, we will use common automatic metrics (to be expanded later) in combination with human testing.

We propose a two-step algorithm for enhancing the Common-Sense Generation model

Step 1: Apply a parsing algorithm to arrange the constraints (the words) in a reasonable order. The purpose of this step is to simplify and logically organize the next step model where we will enforce the order of the words determined in step 1.

Step 2: add and pad mask tokens between the words and apply a generator to complete the masked spans, e.g. {apple, pick, eat} → {pick, apple, eat} → [mask] pick [mask] apple [mask] eat [mask] → A kid picked an apple from the tree and ate it.

According to the needs, the model may generate one, two, or even more words for each mask, and sometimes nothing at all. Furthermore, the model should be able to inflect the roots of constraints (the words) it receives.

In addition, a variety of options for inference will be introduced, each tailored to a specific purpose (i.e. a sentence as coherent as possible, a sentence as creative as possible).

3. Research stages

1. literature review
   Researching various works that perform the task or a similar task for two reasons: The first reason is to find out how this task was done before, to gain inspiration from other people's solutions, to open up my mind to various directions, and to acquire as many tools as possible. Second, it is important to understand what is lacking in the current models, where they go wrong, and how they can be improved or renewed.

   1. Solution planning
   Following the literature review, we developed an approach to solving the problem and define all the key components of the solution. This is accomplished by defining the architecture, choosing an initial model for future fine-tuning, selecting the dataset to be used, Identifying and preventing problems that have arisen in other projects, determining our ambitions for future performance according to what others have accomplished, and more. This stage also included deciding (i) if the solution consist of a single algorithm that works end to end or if it should be divided into smaller tasks; (ii) which compute metrics to use to determine the best parameters?

   2. Implementation
   This step consists of implementing the solution that was theoretically defined in step 2. This phase involves a lot of trial and error. Initially, we implemented a single model that was fine-tuned to perform the task end to end in order to provide us with something comparable. A two-stage solution was also

implemented, which uses two models, as described in more detail below. To accomplish all of this, it is necessary to prepare the data, fine-tune the model, check different parameters and metrics both pertaining to the model and inference, train and test, evaluate analyze and conclude from the results.

3. Project expansion

During the research process, additional directions emerged from the original project. One such direction was to extend the generated sentence as described in the section "Future work".

4. literature review

During the literature review, I focused on two types of articles. First and foremost, there were articles that attempted to propose solutions to the problem. The key to this part is to understand what has been done before, what are the advantages and disadvantages of the solutions, and what makes my solution unique.
A number of interesting and unique works have been published on this topic.

NeuroLogic A*esque Decoding(Ximing Lu2021) presents a decoding algorithm that incorporates heuristic estimates of future cost, drawing inspiration from the A* search algorithm. By using this method, feasible future paths can be planned ahead as opposed to the classical method, which utilizes left-to-right decoding from autoregressive language models. As a result, they are able to "tune" the model toward a sentence that contains the desired words.

NeuroLogic Decoding: (Un)supervised Neural Text Generation(Ximing Lu2021) following on from previous articles about constrained decoding, the idea is to convert the problem into a CNF problem when the constraints are the words. The novelty of their approach is their ability to generalize to arbitrary logical constraints by taking into account the full scope of CNF constraints. As a result, a well-known problem of the model skipping words is solved and it is ensured that all constraints will indeed be met.

Retrieval Enhanced Model for Commonsense Generation(Han Wang 2021) proposing retrieval-enhanced T5 (RE-T5) which equips original T5 with a trainable retriever for selecting prototype sentences based on given concepts. The general idea is retrieve prototype sentence candidates by concept matching and use them as auxiliary input. With the added information that the model receives, they demonstrate that they are able to achieve better results than previous models.

A second type of article focused on suggesting and presenting practical tools for working. These include,

Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer (Colin Raffel, Noam Shazeer 2020) explore the landscape of transfer learning by introducing a unified framework that converts all text-based language problems into a text-to-text format. The authors compare a number of key components of NLP, such as pre-training objectives, architectures, unlabeled data sets, transfer approaches, and other factors on dozens of language understanding tasks. By combining all the insights presented, the T5 model is developed. For both parts of my solution, I used T5 as a core model.

THE CURIOUS CASE OF NEURAL TEXT DeGENERATION (Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, Yejin Choi 2020) propose Nucleus Sampling, a simple but effective method to draw considerably higher quality text out of neural language models than previous decoding strategies. Through using the proposed method as well as the other methods discussed in the article, I was able to adapt my model to generate sentences that emphasize different aspects of language (detailed description provided later).

5. Research overview

The objective of the project is to develop a model capable of automatically creating a forced dataset. As described in the detailed problem definition, our approach to solving the problem is to construct two models, one of which is responsible for arrangement and the other for generation. The literature review did not mention the splitting of the task into two sub-tasks. It will be interesting to explore whether this splitting has any positive effect on the performance of the model (more about that in the result section).

In essence, both models are similar, but they differ in a number of important ways. Each of them is based on T5 but with different data(both based on Common_Gen, however, each of them required a different pre-processing step) that is suitable for its specific problem, with different network parameters that resulted in maximization of the indices and with adjustments which were required for success.

The decision to select T5(Peter J. Liu. 2020), that has been proposed by Google Researches, as the core model for the two algorithms, was due to the architecture and suitability of It's training: This is an encoder-decoder model with each task converted into text-to-text format.

The data I used is "common_gen", as I have already mentioned. Since this data had been published in Hugging Face, it was easy to access and use. Nevertheless, if the existing data were not sufficient, we could easily create new data. The procedure would be as follows: take a text and randomly select three words in each sentence from it. Our input to the model would be this list of words in a messy order, and our output would be the original sentence. This method's power can be demonstrated by the ease with which data augmentation can be performed for the task. Adding critical components like stemming, which clarifies the problem by turning a word into its root, was crucial to accurately defining the problem and its limitations.

A generated text is very difficult to evaluate automatically, however, I have used some very common computation metrics to accomplish this, among them SACREBLEU, 9 versions of ROUGE, 3 versions of GLUE and overlapping count. There are slight differences between each of these methods, but in general they are all interested in identifying a match between the generated sentence and the expected sentence. Having calculated a weighted average, I can begin to determine which of the parameters contribute to the desired result.

Finally, during inference time, I check some common decoding strategy, such as top_p, top_k, and beam search and compared their results. (Ari Holtzman 2020) article summarizes the leading methods in a helpful manner and proposes a new approach that I have also tested. The mentioned above metrics have been obtained for 80 different decoding methods and parameters (There will be a partial display in the results section).

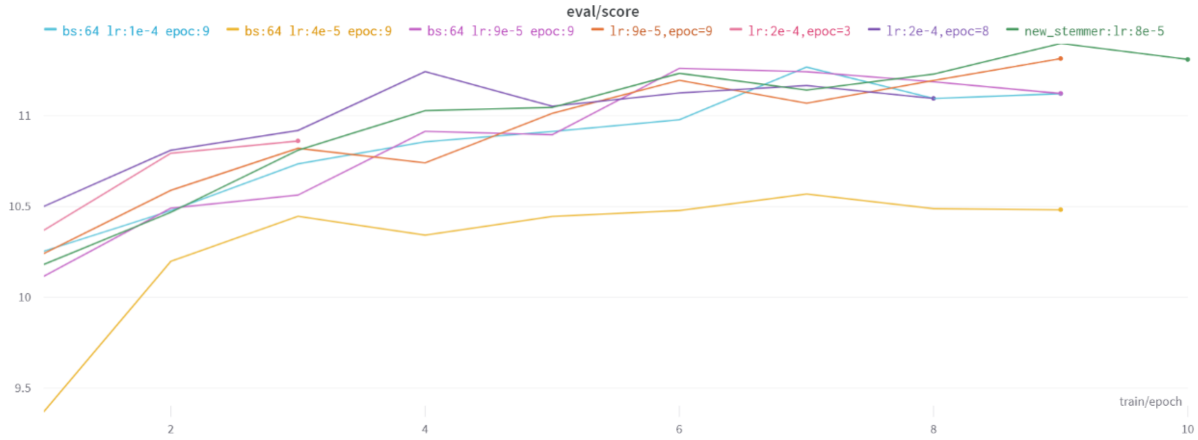
To assess the significance of our idea, we compared the results of our generalized model with a single model trained to perform the task end to end. As you will see in the results section, our model performed significantly better than the single model.

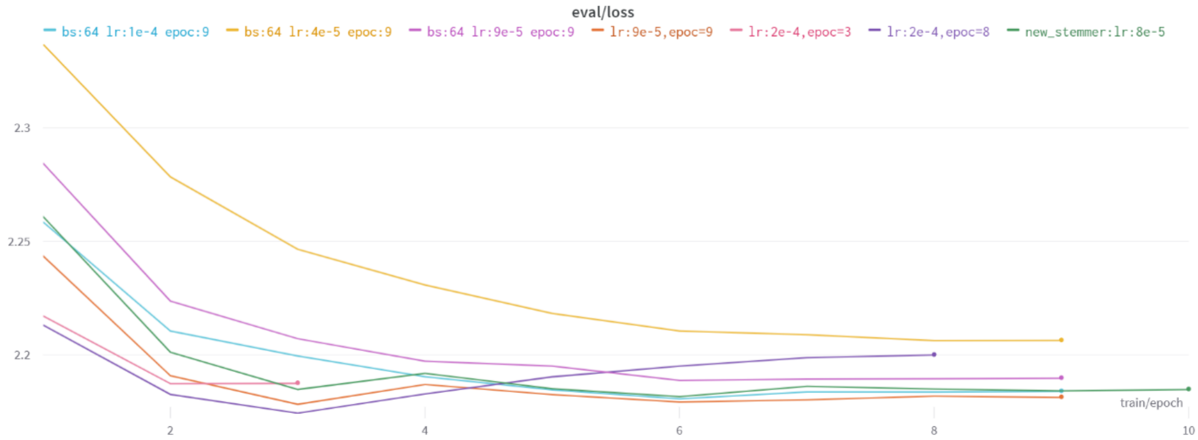
Furthermore, testing the decoding strategy during inference provided some interesting insights, which I will also present in the results section. As a final step, I have added three "heads" to the project, using three different methods (top_p, top_k, beam_search), when each head suggests a different emphasis on the sentence (creativity, clarity, coherence).

6. Research results

It is well known that automatic evaluation of generation tasks is a complex and controversial issue. Due to the fact that such tasks do not have one definitive correct answer (in our case, different sentences can contain the same words), this problem arises. Considering this, 14 metrics were selected to evaluate the results, including SACREBLEU, 9 versions of ROUGE, 3 versions of GLUE and overlapping count.

Using WANDB, which graphically combines all the relevant indices, the final parameters of the model were determined. In the figure below we demonstrate a variety of experiments with different parameters in WANDB (we want a high metric score and a low evaluation loss):

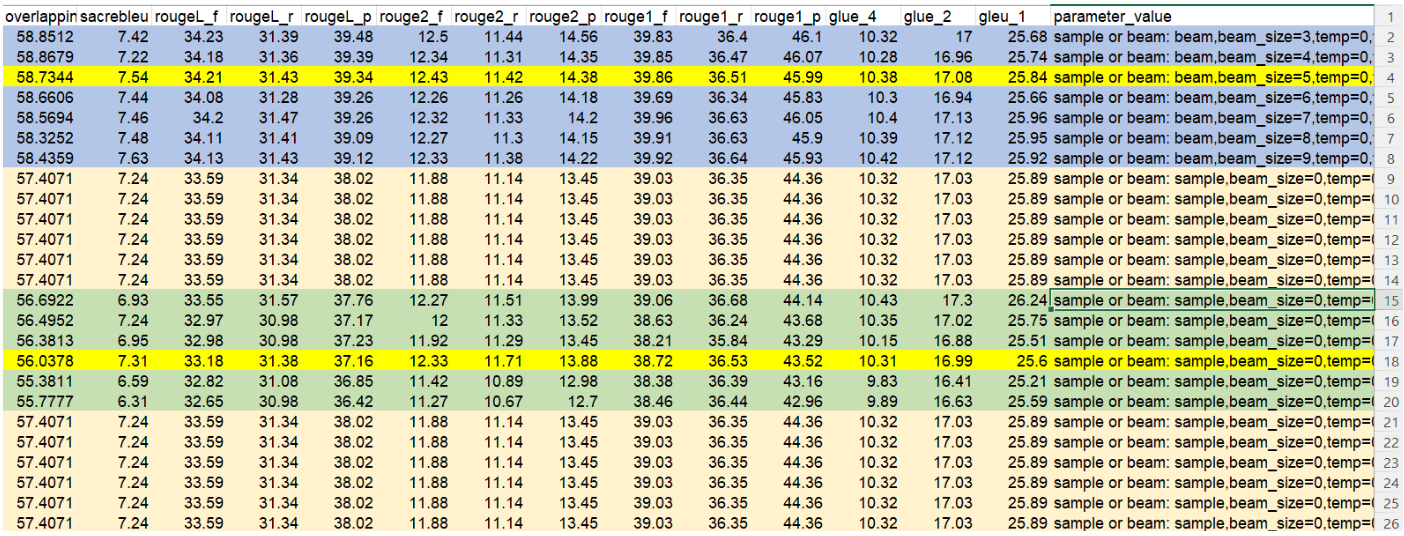




As you can see the green/orange parameters are the best.

For determine the decoding method and it's parameters, I tested 14 metrics for each of the 80 experiments.
A comparison was made between the three methods (top_k, top_p, beam search) and between different parameters of:
The temperature ranges between [0.5,0.6,0.7,0.8,0.9, 1]
A top_k in the range [40,50,60,70,80,90,100]
A top_p between [0.9, 0.92, 0.95, 0.97, 0.99, 0.999]
I then calculated a weighted average of the results and manually verified the top values.

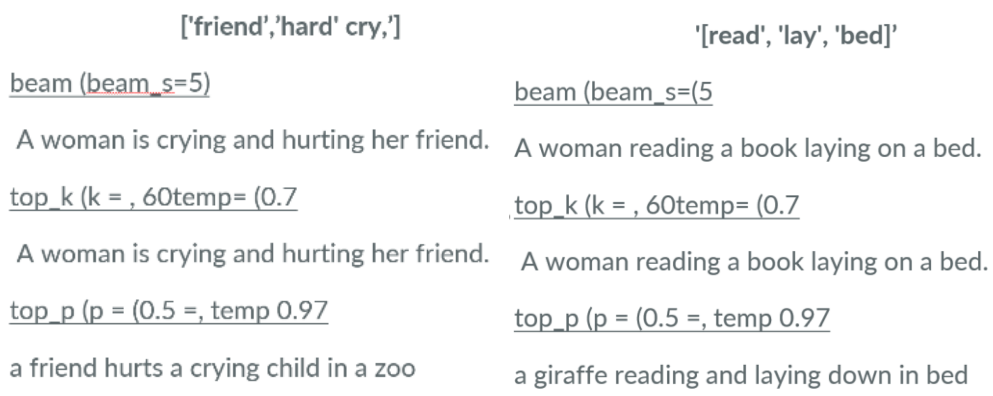
The following is a partial presentation of the information received:



overlappin	sacrebleu	rougeL_f	rougeL_r	rougeL_p	rouge2_f	rouge2_r	rouge2_p	rouge1_f	rouge1_r	rouge1_p	glue_4	glue_2	gleu_1	parameter_value	1
58.8512	7.42	34.23	31.39	39.48	12.5	11.44	14.56	39.83	36.4	46.1	10.32	17	25.68	sample or beam: beam,beam_size=3,temp=0,	2
58.8679	7.22	34.18	31.36	39.39	12.34	11.31	14.35	39.85	36.47	46.07	10.28	16.96	25.74	sample or beam: beam,beam_size=4,temp=0,	3
58.7344	7.54	34.21	31.43	39.34	12.43	11.42	14.38	39.86	36.51	45.99	10.38	17.08	25.84	sample or beam: beam,beam_size=5,temp=0,	4
58.6606	7.44	34.08	31.28	39.26	12.26	11.26	14.18	39.69	36.34	45.83	10.3	16.94	25.66	sample or beam: beam,beam_size=6,temp=0,	5
58.5694	7.46	34.2	31.47	39.26	12.32	11.33	14.2	39.96	36.63	46.05	10.4	17.13	25.96	sample or beam: beam,beam_size=7,temp=0,	6
58.3252	7.48	34.11	31.41	39.09	12.27	11.3	14.15	39.91	36.63	45.9	10.39	17.12	25.95	sample or beam: beam,beam_size=8,temp=0,	7
58.4359	7.63	34.13	31.43	39.12	12.33	11.38	14.22	39.92	36.64	45.93	10.42	17.12	25.92	sample or beam: beam,beam_size=9,temp=0,	8
57.4071	7.24	33.59	31.34	38.02	11.88	11.14	13.45	39.03	36.35	44.36	10.32	17.03	25.89	sample or beam: sample,beam_size=0,temp=	9
57.4071	7.24	33.59	31.34	38.02	11.88	11.14	13.45	39.03	36.35	44.36	10.32	17.03	25.89	sample or beam: sample,beam_size=0,temp=	10
57.4071	7.24	33.59	31.34	38.02	11.88	11.14	13.45	39.03	36.35	44.36	10.32	17.03	25.89	sample or beam: sample,beam_size=0,temp=	11
57.4071	7.24	33.59	31.34	38.02	11.88	11.14	13.45	39.03	36.35	44.36	10.32	17.03	25.89	sample or beam: sample,beam_size=0,temp=	12
57.4071	7.24	33.59	31.34	38.02	11.88	11.14	13.45	39.03	36.35	44.36	10.32	17.03	25.89	sample or beam: sample,beam_size=0,temp=	13
57.4071	7.24	33.59	31.34	38.02	11.88	11.14	13.45	39.03	36.35	44.36	10.32	17.03	25.89	sample or beam: sample,beam_size=0,temp=	14
56.6922	6.93	33.55	31.57	37.76	12.27	11.51	13.99	39.06	36.68	44.14	10.43	17.3	26.24	sample or beam: sample,beam_size=0,temp=	15
56.4952	7.24	32.97	30.98	37.17	12	11.33	13.52	38.63	36.24	43.68	10.35	17.02	25.75	sample or beam: sample,beam_size=0,temp=	16
56.3813	6.95	32.98	30.98	37.23	11.92	11.29	13.45	38.21	35.84	43.29	10.15	16.88	25.51	sample or beam: sample,beam_size=0,temp=	17
56.0378	7.31	33.18	31.38	37.16	12.33	11.71	13.88	38.72	36.53	43.52	10.31	16.99	25.6	sample or beam: sample,beam_size=0,temp=	18
55.3811	6.59	32.82	31.08	36.85	11.42	10.89	12.98	38.38	36.39	43.16	9.83	16.41	25.21	sample or beam: sample,beam_size=0,temp=	19
55.7777	6.31	32.65	30.98	36.42	11.27	10.67	12.7	38.46	36.44	42.96	9.89	16.63	25.59	sample or beam: sample,beam_size=0,temp=	20
57.4071	7.24	33.59	31.34	38.02	11.88	11.14	13.45	39.03	36.35	44.36	10.32	17.03	25.89	sample or beam: sample,beam_size=0,temp=	21
57.4071	7.24	33.59	31.34	38.02	11.88	11.14	13.45	39.03	36.35	44.36	10.32	17.03	25.89	sample or beam: sample,beam_size=0,temp=	22
57.4071	7.24	33.59	31.34	38.02	11.88	11.14	13.45	39.03	36.35	44.36	10.32	17.03	25.89	sample or beam: sample,beam_size=0,temp=	23
57.4071	7.24	33.59	31.34	38.02	11.88	11.14	13.45	39.03	36.35	44.36	10.32	17.03	25.89	sample or beam: sample,beam_size=0,temp=	24
57.4071	7.24	33.59	31.34	38.02	11.88	11.14	13.45	39.03	36.35	44.36	10.32	17.03	25.89	sample or beam: sample,beam_size=0,temp=	25
57.4071	7.24	33.59	31.34	38.02	11.88	11.14	13.45	39.03	36.35	44.36	10.32	17.03	25.89	sample or beam: sample,beam_size=0,temp=	26

As a result of the information provided above, I was able to determine that utilizing different decoding methods during inference time would allow me to achieve good results in different aspects. In all of our tests, top_p has received slightly lower scores than other methods, however, we have observed that the sentences are the most creative. Due to the fact that our metrics examines a different aspect of the sentence, it makes sense that both of these things occur simultaneously. We conclude that using top_p, is most effective when we want creative sentences, while beam_search is most effective when we want always coherent sentences.

here are some differences between the different decoding methods:



['friend’,’hard' cry,’]

beam (beam_s=5)

A woman is crying and hurting her friend.

top_k (k = , 60temp= (0.7

A woman is crying and hurting her friend.

top_p (p = (0.5 =, temp 0.97

a friend hurts a crying child in a zoo

'[read', 'lay', 'bed]’

beam (beam_s=(5

A woman reading a book laying on a bed.

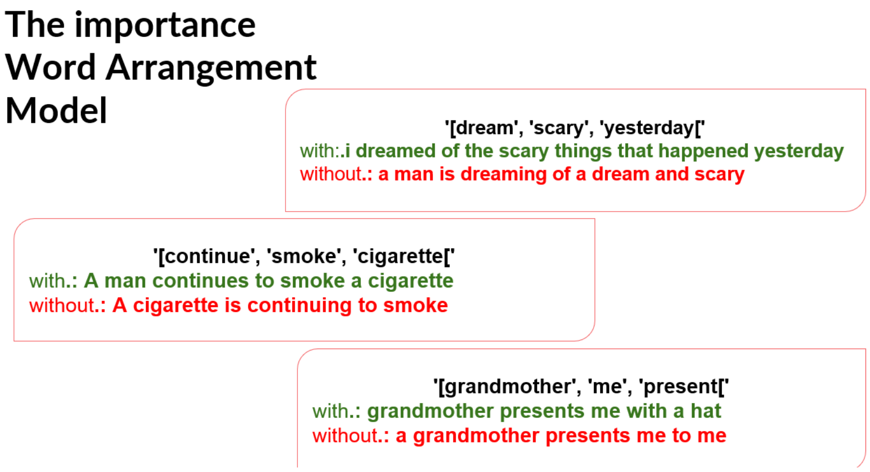
top_k (k = , 60temp= (0.7

A woman reading a book laying on a bed.

top_p (p = (0.5 =, temp 0.97

a giraffe reading and laying down in bed

Remember that our generalized model was compared with a single model that performed the task end-to-end. The following examples illustrate the differences in performance: (green - our generalized model ,red - single end to end model)



The importance
Word Arrangement
Model

'[dream', 'scary', 'yesterday['
with:.i dreamed of the scary things that happened yesterday
without.: a man is dreaming of a dream and scary

'[continue', 'smoke', 'cigarette['
with.: A man continues to smoke a cigarette
without.: A cigarette is continuing to smoke

'[grandmother', 'me', 'present['
with.: grandmother presents me with a hat
without.: a grandmother presents me to me

To demonstrate the effectiveness of the model, we conducted a field test. There are several examples in which the model receives a list of constraints (words) and generated human-like coherent texts from this constraints :

'[drive', 'road', 'car['
A man driving a car down a road.

'[come', 'party', 'girl['
A girl is coming to a party.

'[soldier', 'Ukraine', 'war['
a soldier during a military war in Ukraine

'[friend', 'cry', 'hurt['
A woman is crying and hurting her friend.

'[London', 'entertainment', 'friends['
a group of friends enjoying entertainment in the city of London

'[hard', 'work', 'why['
work hard to find out why.

7. Future work

This work provides a method for generating constraint sentences based on a set of words. One interesting question is how can this idea be generalized? Can a model be developed that will accept not only a set of words, but also a sentence, and insert the words within? A second approach would be to add structural constraints, The model should be able to generate sentences of a specific structure (nouns, verbs, etc.), questions, arguments, etc. It would also be interesting to consider the possibility of implementing our model in larger problems. Consider the following example: given a dataset that is prone to social discrimination, construct a generalized model based on our model that generate a new dataset, which is free from social discrimination. Other future directions can be considered.

8. Bibliography

- Ximing Lu, Sean Welleck, Peter West, Liwei Jiang, Jungo Kasai, Daniel Khashabi, Ronan Le Bras, Lianhui Qin, Youngjae Yu, Rowan Zellers, Noah A. Smith, Yejin Choi. 2021. NeuroLogic A*esque Decoding: Constrained Text Generation with Lookahead Heuristics. arXiv:2112.08726v1 [cs.CL]
- Ximing Lu, Peter West, Rowan Zellers, Ronan Le Bras, Chandra Bhagavatula, Yejin Choi. 2021. NeuroLogic Decoding: (Un)supervised Neural Text Generation with Predicate Logic Constraints. NAACL 2021
- Han Wang, Yang Liu, Chenguang Zhu, Linjun Shou, Ming Gong, Yichong Xu, Michael Zeng 2021. Retrieval Enhanced Model for Commonsense Generation
- Nishtha Madaan, Srikanta Bedathur, Diptikalyan Saha. 2022. Plug and Play Counterfactual Text Generation for Model Robustness. arXiv:2206.10429v1 [cs.CL] 21 Jun 2022
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer
- Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, Yejin Choi 2020. THE CURIOUS CASE OF NEURAL TEXT DeGENERATION