Read the paper at https://arxiv.org/abs/1902.04094 and implement a conditional generative model on top of it.

Paper mentioned in link represent BERT as Markov random field language model. This formulation gives way to sample sentences from BERT. Author represent BERT as a combination of a Markov random field language model (MRF-LM) with pseudo loglikelihood training. This formulation automatically leads to a sampling procedure based on Gibbs sampling.

In Gibbs sampling, we start with a random initial state X^0, which we initialize to be an all-mask sequence, i.e., ([MASK] ; … ; [MASK]). At each iteration i, we sample the position t^i uniformly at random from {1; … ; T} and mask out the selected location, i.e., x^i_{ti} = [MASK] and BERT will generate that masked word.

Approach:

In Gibbs sampling, for any sentence we start with all-mask sequence (kind of noise) and at the end we end up generating full sentence based on that all-mask sequence. As we are doing random sampling this process will generate different sentences for same all-mask sequence input so In order to sample conditional sentences we have to add that condition in all-mask sequence and BERT will generate sentence based on that condition.

In order to generate conditional sentences initial state x^0 is all-mask sequence with some [mask] replaced with "condition" i.e., ([mask] ; … ; condition; … ; [mask]).At each iteration i, we also have to take care that sample position t^i will not replace "condition" with [mask].

Implementation:

In order to implement above mentioned approach I used code available here https://github.com/nyu-dl/bert-gen and attached with paper. I have modified some of the existing code and added new code as per requirements. Code that I have added to generate conditional sentences:

Below functional will initialize initial sequence based on given condition. Function will introduce condition in initial sentence uniformly at random:

```
def get_init_text(seed_text,inp_txt, max_len, batch_size = 1,
rand_init=False):
    """ Get initial sentence by padding seed_text with condition
and mask of lenght max_len """

    batch = list()
    inp_len = len(inp_txt)
    seed_len = 1

    # randomly introduce condition on initial sequence
    rand_ind = np.random.randint(0, max_len - inp_len + 1)
    for ind in range(batch_size):
        temp = seed_text + [MASK] * rand_ind + inp_txt + [MASK]
* (max_len - rand_ind - inp_len) + [SEP]
        batch.append(temp)

    return [tokenize_batch(batch),rand_ind]
```

Below function will collect all the indices that we can mask and not contain indices of condition:

```
def getLeftInd(inp_ind,inp_len,max_len):
    """ Generate indices that we can mask """
    ind_flag = [False]*max_len
    for i in range(inp_len):
        ind_flag[inp_ind + i] = True

    left_ind = list()
    for i in range(max_len):
        if not ind_flag[i]:
            left_ind.append(i)

    return left_ind
    left_len = len(left_ind)
    kk = np.random.randint(0, left_len)
    return left_ind[kk]
```

Below function will choose index uniformly at random for masking:

```
def getRandInd(left_ind):
    """ left_ind contains indices of sentence except
"condition". function will randomly choose index for mask. """
    left_len = len(left_ind)
    kk = np.random.randint(0, left_len)
    return left_ind[kk]
```

Also I have modified function “parallel_sequential_generation(…)” available in GitHub repository (link mentioned above) to generate conditional sentences.

Generations:

We can control sentence generation process by modifying variables mentioned below:

```
In [0]: n_samples = 20
        batch_size = 2
        sentence_len = 15
        max_iter = 500
        condition = "really happy"

        file_name = "/content/drive/My Drive/Colab Notebooks/generated_language.txt"
```

By modifying “condition” variable we can generate sentences based on different – different conditions. “sentence_len” indicate length of the sentence that we want to generate. In normal computer without GPU it will take approx. 23 minutes to generate 20 sentences of length 15 with 500 iterations. Below are the 20 sentences generated with condition “really happy”:

```
i was really happy about the change . i had never been to this festival before .
they look really happy now , too , some really happy , but mostly not happy .
" oh hello , madaug ! i am really happy just having you back .
and if there was anything to ask , i was really happy to see him tonight .
that boy really loved me . then there were two . and i was really happy .
that was when my parents hugged me . oh god , i was just really happy .
the painting presents white silhouettes to symbolize how there will be really happy marriages .
pity i cannot keep my eyes off him , but he ... looks really happy inside .
" really happy to see you again . " sean loved the sound of the bear .
" really happy , " he muttered . " so happy . so damn happy . "
not really happy either , although i hear it a lot . he checks his watch .
never really happy with herself , she was married to a man from the far east .
opening the door i exclaim , " how are you looking ? " really happy .
by the minute , you can see videos in high definition of people being really happy .
" your mom is really kind , " my mom said . she sounded really happy .
embarrassed , i thought back to north central . mom and dad had been really happy .
" that was my daddy . " jeff said , admitting they were really happy together .
like this was a sign that we were both really , really , really happy together .
it was more happy than anything else , and i was actually really happy about that .
" i still think the same thing . " he seemed ... really happy , actually .
```