[Temperature and Top\_p in ChatGPT](https://medium.com/@basics.machinelearning/temperature-and-top-p-in-chatgpt-9ead9345a901" \l ":~:text=What%20is%20Temperature%20in%20ChatGPT,ChatCompletion%20function%20(among%20others).)

[Hanane D.](https://medium.com/@basics.machinelearning?source=post_page-----9ead9345a901--------------------------------)·

What is Temperature in ChatGPT API?

The temperature parameter is a hyperparameter used in language models (like GPT-2, GPT-3, BERT) to control the randomness of the generated text. It is used in the ChatGPT API in the ChatCompletion function (among others).

It controls how much the model should take into account low-probability words when generating the next token in the sequence.

The parameters control the degree of randomness or creativity (depending on the task) in the generated output by modifying the probabilities of activation functions like softmax function:

softmax(x) = exp(x/temperature) / sum(exp(x/temperature))

A lower value of the temperature parameter will lead to a more predictable and deterministic output, while a higher value will produce a more random and surprising output.

According to OpenAI documentation, In gpt-3.5-turbo temperature values are between 0 and 2. The default value is set to 1.

This value can be adjusted by the user when generating the text to fit the desired output.

Let’s see how it works. In the following, I will use the ChatGPT API openai.

*Have a look at the current offer — “*[*Technical Indicators Using Python and TA-Lib*](https://machinelearning-basics.com/ebook-technical-indicators/)*”, more than 25 Indicators implemented from scratch in Python, with in-depth interpretation.*

ChatGPT API — Python

If you don’t know how to set up and use this API, have a look at my latest article : [ChatGPT API Using Python](https://machinelearning-basics.com/chatgpt-api-using-python/).

Temperature = 1: Default value

%%time  
response=openai.ChatCompletion.create(  
 model='gpt-3.5-turbo',  
 messages=[{"role":"user",  
 "content":"What are the different examples of activation functions used in deep learning models, their definition and their formulas?"}  
 ]  
)  
  
first\_response = response.choices[0].message.content  
print(first\_response)

I get a list of the activation functions (not complete but representative), with a short definition and the formula. It takes 56.4s to have the answer :

*You need to know that the API is very slow compared to the GUI chat (Have a look at the end of the article “API Performance”).*

1. Sigmoid Function:  
The sigmoid function takes any range real number and returns a value between 0 and 1. It is one of the most commonly used activation functions in deep learning.   
Formula: f(x) = 1/(1+e^-x)  
  
2. ReLU Function:  
ReLU stands for Rectified Linear Unit. It is a non-linear activation function, which returns the input if it's positive, and if not, it returns zero.   
Formula: f(x) = max(0,x)  
  
3. Tanh Function:  
The Tanh function is a popular activation function that is symmetric around the origin, which means it returns values between -1 and 1.   
Formula: f(x) = (e^x - e^-x) / (e^x + e^-x)  
  
4. Leaky ReLU Function:  
Leaky ReLU is a variant of the ReLU function, which allows a small, non-zero gradient when the input is negative, solving the "dying ReLU" problem.   
Formula: f(x) = max(0.01x, x)  
  
5. Softmax Function:  
The Softmax function is another commonly used activation function. It returns an output in the range of [0,1] and ensures that the sum of all output values is 1, which makes it ideal for multi-class classification problems.  
Formula: f(x) = e^xi / ∑(e^xj)  
  
Wall time: 56.4 s

Temperature = 0

%%time  
response=openai.ChatCompletion.create(  
 model='gpt-3.5-turbo',  
 messages=[{"role":"user",  
 "content":"What are the different examples of activation functions used in deep learning models?"}  
 ],  
 temperature = 0  
)  
  
second\_response = response.choices[0].message.content  
print(second\_response)

I get more straightforward answers, and it takes 39.9s (less than the first call), but the first call has given valid answers too.

1. Sigmoid Function: The sigmoid function is a commonly used activation function in deep learning models. It maps any input value to a value between 0 and 1. The formula for the sigmoid function is:  
f(x) = 1 / (1 + e^-x)  
  
2. ReLU Function: The Rectified Linear Unit (ReLU) function is another popular activation function used in deep learning models. It maps any input value less than 0 to 0 and any input value greater than 0 to the input value itself. The formula for the ReLU function is:  
f(x) = max(0, x)  
  
3. Tanh Function: The hyperbolic tangent (tanh) function is similar to the sigmoid function, but it maps any input value to a value between -1 and 1. The formula for the tanh function is:  
f(x) = (e^x - e^-x) / (e^x + e^-x)  
  
4. Softmax Function: The softmax function is commonly used in the output layer of deep learning models for classification tasks. It maps any input value to a probability distribution over multiple classes. The formula for the softmax function is:  
f(x\_i) = e^x\_i / sum(e^x\_j) for j = 1 to n  
where n is the number of classes and x\_i is the input value for class i.  
  
5. Leaky ReLU Function: The Leaky ReLU function is a modified version of the ReLU function that allows for a small non-zero gradient for negative input values. The formula for the Leaky ReLU function is:  
f(x) = max(0.01x, x)  
  
Wall time: 39.9 s

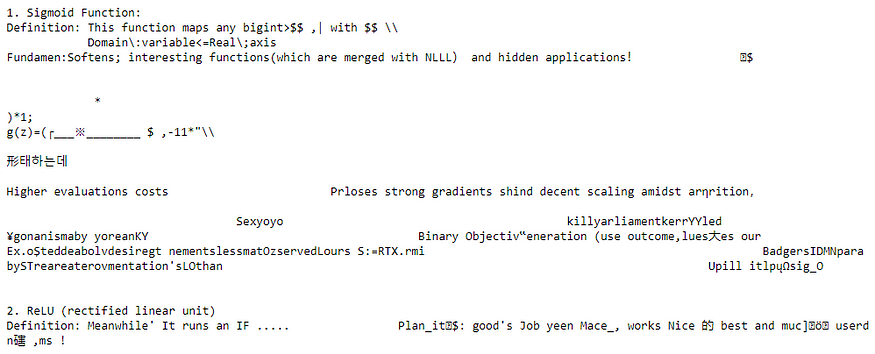
Temperature = 1.9

*With Temperature = 2, I get a Gateway timeout, so I used 1.9*

%%time  
response=openai.ChatCompletion.create(  
 model='gpt-3.5-turbo',  
 messages=[{"role":"user",  
 "content":"What are the different examples of activation functions used in deep learning models?"}  
 ],  
 temperature = 1.9  
)  
  
third\_response = response.choices[0].message.content  
print(response.choices[0].message.content)

The answer nothing to do with the question and the request takes a lot of time 4 min 54s:

Here is a quick overview of the answer:



Now let’s have a look at another parameter called top\_p which is also used in the ChatGPT API.

What is top\_p in ChatGPT API?

The top\_p parameter can also be used to control the randomness of the outputs.

*OpenAI documentation recommends modifying either temperature or top\_p, but not both.*

Top\_p sampling is also called nucleus sampling, in which a probability threshold is set (Default value =1 in the API). This threshold represents the proportion of the probability distribution to consider for the next word. In other words, It consists of selecting the top words from the probability distribution, having the highest probabilities that add up to the given threshold.

For example, if we set a top\_p of 0.05, it means that the model, once it generated the probability distribution, will only be considering the tokens that have the highest probabilities, and sum up to 5%. Then the model will be randomly selecting the next token among these 5% tokens, according to its likelihood.

The top\_p sampling is highly correlated to the quality and the size of the dataset used to train the model. In Machine learning subjects, as there are huge datasets with good quality, the answers are not that different when modifying the value of top\_p.

You can run this code, and see how the outputs are different:

import time  
  
top\_ps = [0, 0.1,0.5,0.8,1 ]  
start\_time = time.time()  
  
for top\_p\_v in top\_ps:  
 print(top\_p\_v)  
 response=openai.ChatCompletion.create(  
 model='gpt-3.5-turbo',  
 messages=[{"role":"user",  
 "content":"What are the different examples of activation functions used in deep learning models?"}  
 ],  
 top\_p=top\_p\_v  
 )  
   
 call\_time=time.time()  
 print("duration", call\_time-start\_time)  
 start\_time=call\_time  
   
 print(response.choices[0].message.content)  
 print("")

I give here just the first (top\_p=0) and the last (top\_p=1) outputs:

As you can see there are more words in top\_p=1 than the ones in top\_p=0. But fundamentally, as said before, as the training datasets of machine learning are huge with a good quality, we end up with good results in both cases.

0  
duration 65.72348475456238

There are several activation functions used in deep learning models, including:  
1. Sigmoid function: It maps any input value to a value between 0 and 1. It is commonly used in binary classification problems.  
  
2. ReLU (Rectified Linear Unit) function: It maps any input value to 0 if it is negative, and to the same value if it is positive. It is commonly used in convolutional neural networks (CNNs).  
  
3. Tanh (Hyperbolic Tangent) function: It maps any input value to a value between -1 and 1. It is commonly used in recurrent neural networks (RNNs).  
  
4. Softmax function: It maps any input value to a probability distribution over multiple classes. It is commonly used in multi-class classification problems.  
  
5. Leaky ReLU function: It is similar to the ReLU function, but it allows a small positive slope for negative input values. It is used to prevent the dying ReLU problem.  
  
6. ELU (Exponential Linear Unit) function: It is similar to the ReLU function, but it allows a small negative slope for negative input values. It is used to prevent the dying ReLU problem and improve the performance of deep neural networks.  
  
7. Swish function: It is a recently proposed activation function that is similar to the sigmoid function, but it has a smoother curve and can improve the performance of deep neural networks.

1  
duration 66.14234447479248

There are several examples of activation functions used in deep learning models, including:  
1. Sigmoid function: The sigmoid function is a common activation function, which maps the input to a value between 0 and 1. It is primarily used in binary classification problems.  
  
2. ReLU: The Rectified Linear Unit (ReLU) is one of the most widely used activation functions in deep learning. It maps the input to 0 if negative or to the same value if positive.  
  
3. Tanh function: Tanh is a hyperbolic tangent function that maps the input to a value between -1 and 1. It is used in classification and regression problems.  
  
4. Softmax function: The softmax function is a popular activation function used at the output layer of neural networks for multi-class classification.  
  
5. LeakyReLU: LeakyReLU is an upgraded version of ReLU, where the negative values have a small non-zero positive slope to prevent dead neurons.  
  
6. SeLU: Scaled exponential linear units (SeLU) is an activation function designed to improve the performance of deep neural networks. It combines the benefits of ReLU and tanh without being affected by vanishing gradients.  
  
7. Swish: Swish is a new activation function, which is reported to outperform traditional functions because of its smoothness, having a thresholding effect, and computational efficiency.

API Performance

Note that the API performance is not that good, I have 50 s average time when I make a call. The GUI is way faster. I’m sending small requests, with average prompt\_tokens of 50 and get completion\_tokens around 300.

It seems that this slowness could only be observed on the free version. The ChatGPT Plus (non free version) is faster. I didn’t test it, so I cannot assure you that is the case.

Summary

The temperature controls the correctness of the answer. Higher values of this parameter will lead to more random and surprising answers, while low values will generate the most probable outputs. I usually use the default value of temperature, however if the results start to become too varied, I lower the value.

The top\_p parameter can also be used to impact the outputs. Lower values will give shortened answers. One cannot modify both parameters at the same time.

To note that there is also another way to let ChatGPT be more creative: We can instruct ChatGPT the *tone*it can use to answer our questions. How to influence the tone of ChatGPT answers will be explained in another article.

*You can have a look on the available resources on my blog:*

[*https://machinelearning-basics.com/ressources/*](https://machinelearning-basics.com/ressources/)

**Appendix: Explanations from Shashank regarding temperature & top\_p**

**Temperature in a chatgpt**

In a GPT model, the temperature is a hyperparameter that controls the randomness of the output generated by the model.  
Specifically, the temperature controls the probability distribution over the tokens in the output. When the temperature is higher, the model is more likely to generate output that is diverse and creative, but may also be less coherent and less focused on the input. When the temperature is lower, the model is more likely to generate output that is focused and coherent, but may also be less diverse and less creative.

The temperature is set when the model is trained, and it can also be adjusted when the model is used to generate output. For example, you can set the temperature to a higher value to generate more diverse and creative output, or to a lower value to generate more focused and coherent output.

In the context of a chatbot, the temperature can be used to control the randomness of the responses generated by the bot. For example, you can set the temperature to a higher value to make the bot more creative and less predictable, or to a lower value to make the bot more focused and coherent in its responses.  
  
**what is top\_p**  
In a GPT model, the top\_p parameter is a hyperparameter that controls the diversity of the output generated by the model.

Specifically, the top\_p parameter controls the sampling process used to generate the output. When top\_p is set to a value between 0 and 1, the model samples tokens from the output distribution using a cumulative probability threshold. This means that the model selects the tokens that have a cumulative probability greater than or equal to top\_p, and discards the rest.  
The effect of top\_p is similar to that of the temperature parameter, in that it controls the diversity of the output generated by the model. When top\_p is set to a higher value, the model is more likely to generate output that is diverse and creative, but may also be less coherent and less focused on the input. When top\_p is set to a lower value, the model is more likely to generate output that is focused and coherent, but may also be less diverse and less creative.

In the context of a chatbot, the top\_p parameter can be used to control the diversity of the responses generated by the bot. For example, you can set top\_p to a higher value to make the bot more creative and less predictable, or to a lower value to make the bot more focused and coherent in its responses.