Foundation model parameters: decoding and stopping criteria

You can set parameters to control how the model generates output in response to your prompt. Set decoding parameters to adjust how the output text is generated. Set stopping criteria parameters to specify when the model should stop generating output.

Decoding

Decoding is the process that a model uses to choose the tokens in the generated output.

Choose one of the following decoding options:

- **Greedy decoding**: Selects the token with the highest probability at each step of the decoding process.

  Greedy decoding produces output that closely matches the most common language in the model's pretraining data and in your prompt text, which is desirable in less creative or fact-based use cases. A weakness of greedy decoding is that it can cause repetitive loops in the generated output.

  Greedy decoding does not always generate the same output in consecutive prompts. To get consistent outputs across multiple prompts, use sampling decoding and specify the same number for the random seed parameter in each prompt.

- **Sampling decoding**: Offers more variability in how tokens are selected.

  With sampling decoding, the model *samples* tokens, meaning the model chooses a subset of tokens, and then one token is chosen randomly from this subset to be added to the output text. Sampling adds variability and randomness to the decoding process, which can be desirable in creative use cases. However, with greater variability comes a greater risk of incorrect or nonsensical output.

More options for sampling decoding

When you choose *Sampling* decoding, more parameters are available that you can use to adjust how the foundation model chooses tokens to sample. The following parameters work together to influence which tokens are sampled:

- *Temperature sampling* flattens or sharpens the probability distribution over the tokens to be sampled.
- *Top-k sampling* samples tokens with the highest probabilities until the specified number of tokens is reached.
- *Top-p sampling* samples tokens with the highest probability scores until the sum of the scores reaches the specified threshold value. (Top-p sampling is also called *nucleus sampling*.)

Parameter	Supported values	Default	Use
Temperature	Floating-point number in the range 0.0 (same as greedy decoding) to 2.0 (maximum creativity)	0.7	Higher values lead to greater variability
Top K	Integer in the range 1 to 100	50	Higher values lead to greater variability
Top P	Floating-point number in the range 0.0 to 1.0	1.0	Unless you change the value, this setting is not used

The Temperature, Top K, and Top P settings are not applicable when you choose to use greedy decoding.

Example of adjusting sampling decoding settings

In this example, the foundation model already generated the output text `I took my dog` and now the model is choosing the next token.

To find the best choice, the model calculates a discrete probability distribution over the possible tokens. With this type of distribution, each token is assigned a decimal point probability score between 0 and 1 where the scores add up to 1.

In a real scenario, there might be hundreds of possible tokens. In this example, the choices include only five tokens, which are shown here in the context of typical sentences:

I took my dog...

- *for* a walk.
- *to* the vet.
- *with* me.
- *and* my cat on vacation.
- *by* the collar.

Top K and Top P represent two different methods for choosing the tokens to sample.

Both methods begin by ordering the choices from most-to-least probable. The following table lists the tokens and their fictional probability scores in order.

Token	Probability score
for	0.4
to	0.25
with	0.17
and	0.13
by	0.05

Top K example

Top K specifies how many tokens to sample. For example, if you set Top K to 3, then only the first three tokens in the list are sampled: *for*, *to*, and *with*.

Token	Probability score
for	0.4
to	0.25
with	0.17
and	0.13
by	0.05

Note: A Greedy decoding setting is equivalent to Top K = 1.

Top P example

Top P specifies the cumulative probability score threshold that the tokens must reach.

For example, if you set Top P to 0.6, then only the first two tokens, *for* and *to*, are sampled because their probabilities (0.4 and 0.25) add up to 0.65. (As shown in this example, it is okay for the sum to exceed the threshold.)

Token	Probability score
for	0.4
to	0.25
with	0.17
and	0.13
by	0.05

0.4 + 0.25 = 0.65

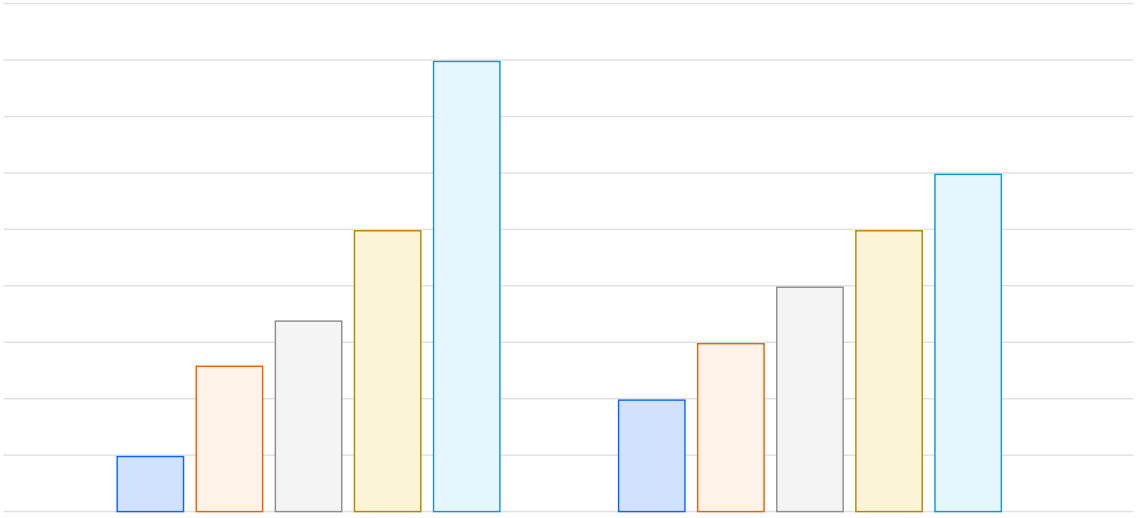
Top P is not used unless you set the Top P parameter value to something other than the default value of 1. Using Top P and Top K together can be a useful way to filter out tokens with extra low probability scores. When both parameters are specified, Top K is applied first.

For example, you might set Top K to 5 and Top P to 0.8. The Top K setting samples all 5 tokens, and then Top P limits the sampled tokens to *for*, *to*, and *with* because their probabilities reach the probability score threshold of 0.8 (0.4 + 0.25 + 0.17 = 0.82).

When both settings are specified, any tokens below the cutoff that is set by Top K are considered to have a probability of zero when Top P is computed. For example, if Top K is set to 2 and Top P to 0.8, then only *for* and *to* are sampled. The token *with* is not sampled because the probability scores for *with*, *and*, and *by* are reset to 0.

Temperature example

The temperature setting affects the shape of the probability distribution that is used when sampling tokens.



Low temperature **High temperature**

Low temperatures amplify the probability differences between tokens. More-likely terms have much higher scores relative to less-likely terms. As a result, terms that are similar to terms in the model's training data or your prompt input will probably be sampled. Use a lower temperature value when you want more dependable output.

High temperatures result in token probabilities that are closer to one another. As a result, unusual terms have a better chance of being sampled. Use a higher temperature value when you want to increase the randomness and variability or the output, such as when you want creative output. Remember, randomness can also lead to inaccurate or nonsensical output.

For example, when a high temperature value such as 2 is applied, the probability scores of the tokens in this example might be closer to one another, as shown in the following table.

Token	Scores with temperature = 2
for	0.3
to	0.25
with	0.2
and	0.15
by	0.10

When Top P is set to 0.8, temperature affects the sample tokens as follows:

- With a high temperature, the top four tokens (*for*, *to*, *with*, and *and*) are sampled because their scores (0.3 + 0.25 + 0.2 + 0.15 = 90) total 90.
- With a low temperature, only the top three tokens (*for*, *to*, and *with*) are sampled because their scores (0.4 + 0.25 + 0.17 = 82) total 82.

Token	Score with temperature = 0.7	Scores with temperature = 2
for	0.4	0.3
to	0.25	0.25
with	0.17 sum = 0.82	0.2
and	0.13	0.15 sum = 0.9
by	0.05	0.10

When a high temperature value is used, it takes more tokens to reach the threshold that is set by Top P. The extra token that gets sampled has the lowest score, which means that the token is a more unusual choice.

Random seed

When you submit the same prompt to a model multiple times with sampling decoding, the model usually generates different text each time. This variability is the result of intentional pseudo-randomness that is built into the decoding process.

Random seed refers to the number that is used to start the random number generator that the model uses to randomize its token choices. If you want to remove this intentional randomness as a variable from your experiments, you can pick a number and specify that same number each time you run the experiment.

- **Supported values:** Integer in the range 1 to 4,294,967,295
- **Default:** Itself randomly generated
- **Use:** To produce repeatable results, set the same random seed value every time.

Repetition penalty

If the generated output for your chosen prompt, model, and parameters consistently contains repetitive text, you can try adding a *repetition penalty*. When set, the penalty lowers the probability scores of tokens that were recently used so that the model is less likely to repeat them. A higher value leads to more diverse and varied output.

- Supported values: Floating-point number in the range 1.0 (no penalty) to 2.0 (maximum penalty)
- Default value: 1.0
- Use: The higher the penalty, the less likely it is that the result will include repeated text.

Stopping criteria

Text generation stops after the model considers the output to be complete, a stop sequence is generated, the maximum token limit is reached, or the model generation time limit is reached.

Model generation stops when the time limit for the generation request is reached. The default time limit is 10 minutes (5 minutes for Lite plans). You can specify a shorter time limit when you submit an inference request by using the watsonx.ai API.

You can affect the length of the output that is generated by the model in the following ways: specifying stop sequences and setting Min tokens and Max tokens.

Stop sequences

A *stop sequence* is a string of one or more characters. If you specify stop sequences, the model will automatically stop generating output after one of the stop sequences that you specify appears in the generated output.

For example, one way to cause a model to stop generating output after just one sentence is to specify a period as a stop sequence. That way, after the model generates the first sentence and ends it with a period, output generation stops.

Choosing effective stop sequences depends on your use case and the nature of the generated output that you expect.

- Supported values: 0 to 6 strings, each no longer than 40 tokens
- Default value: No stop sequence
- Use: Follow these tips for using stop sequences:
    - Stop sequences are ignored until after the number of tokens that are specified in the Min tokens parameter are generated. Keep this relationship in mind when you set the minimum token parameter value.
    - If your prompt includes examples of input-and-output pairs, be sure to include one of the stop sequences in the sample output in your examples.

Minimum and maximum new tokens

If the output from the model is too short or too long, try adjusting the parameters that control the number of generated tokens:

- The *Min tokens* parameter controls the minimum number of tokens in the generated output
- The *Max tokens* parameter controls the maximum number of tokens in the generated output
- Supported values: The maximum number of tokens that are allowed in the output differs by model.
- Defaults:
    - Min tokens: 0
    - Max tokens: 200
- Use:
    - Min tokens must be less than or equal to Max tokens.
    - The cost of using foundation models in IBM watsonx.ai is based on use, which is partly related to the number of tokens that are generated. Specifying the lowest value for Max tokens that works for your use case is a cost-saving strategy.