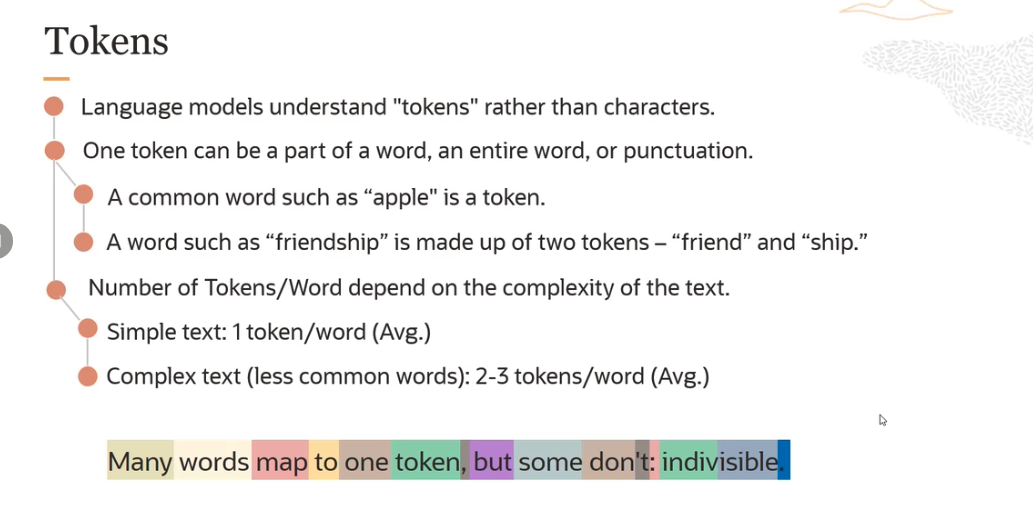
**What are tokens?**

Large language models understand tokens rather than characters.

One token can be part of a word. It could be an entire word or it could be even a punctuation symbol. A common word such as apple is a token. Another word such as friendship is made up of two tokens, friend and ship.



So this is an example of how large language models take text as input. And then, they tokenize that input. They understand tokens rather than characters.

**Pretrained Generation Models in Generative AI:**

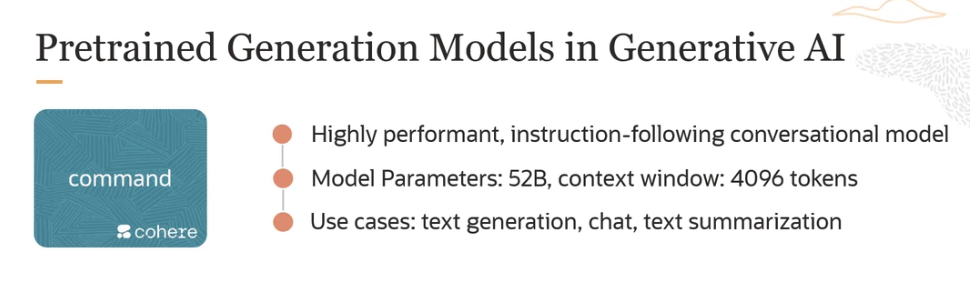
**1.** **Command model** from cohere:

- It's the highly performant instruction following conversational model.

- It has 52 billion parameters and could be used for text generation, text summarization, or chat use cases.

- It has a context window of 4,096 tokens. A model's context window refers to the number of tokens it's capable of processing at one time.

- It's the sum of input and output tokens for that particular model.



Another model from cohere

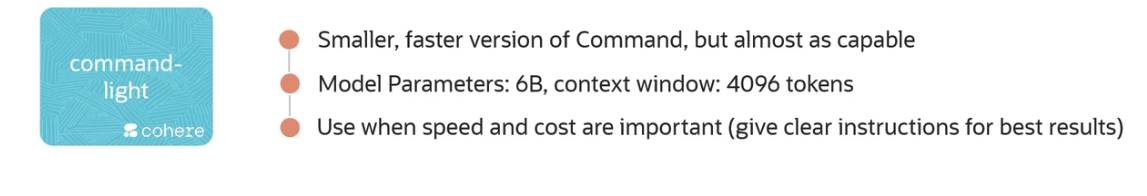
**2. Command light :**

- Command light basically is the smaller, faster version of the command model but almost as capable.

- It has far fewer parameters, 6 billion parameters.

- The context window is the same(4,096 tokens). And should use it when speed and cost are important.

- Because this model is light, for best results, give the model clear instructions. The more specific your prompt, the better this model performs.



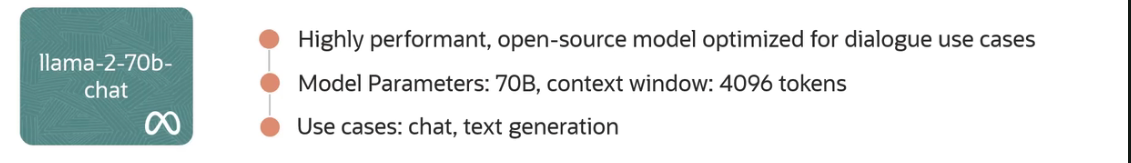
Final model

**3. Llama 2-70 billion chat model:**

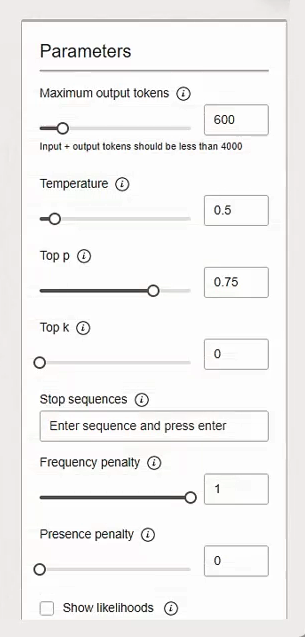
- The llama chat model comes in three sizes, 13 billion parameters or 7 billion parameters, 13 billion parameters and 70 billion parameters. The one we support in OCI is the 70 billion parameter model.

- This is highly performant model. It's an open source model.

- This model is optimized for dialogue use cases, it's used for chat. It could be used for text generation as well.



**Generation Model PARAMAETERS:**

****

**1. Maximum Output Tokens:** This is the max number of tokens model generates per response. In case of OCI, the limit is 4,000 tokens.

**2. Temperature:** This basically determines how creative the model should be. it's close to prompt engineering in controlling the output of generation models.

**3. Top p and Top k:** These are two additional ways to pick the output tokens besides temperature.

**4. Penalty:** They assign a penalty when a token appears frequently and produces less repetitive text.

**5. Show likelihood:** It determines how likely it would be for a token to follow the current generated token.

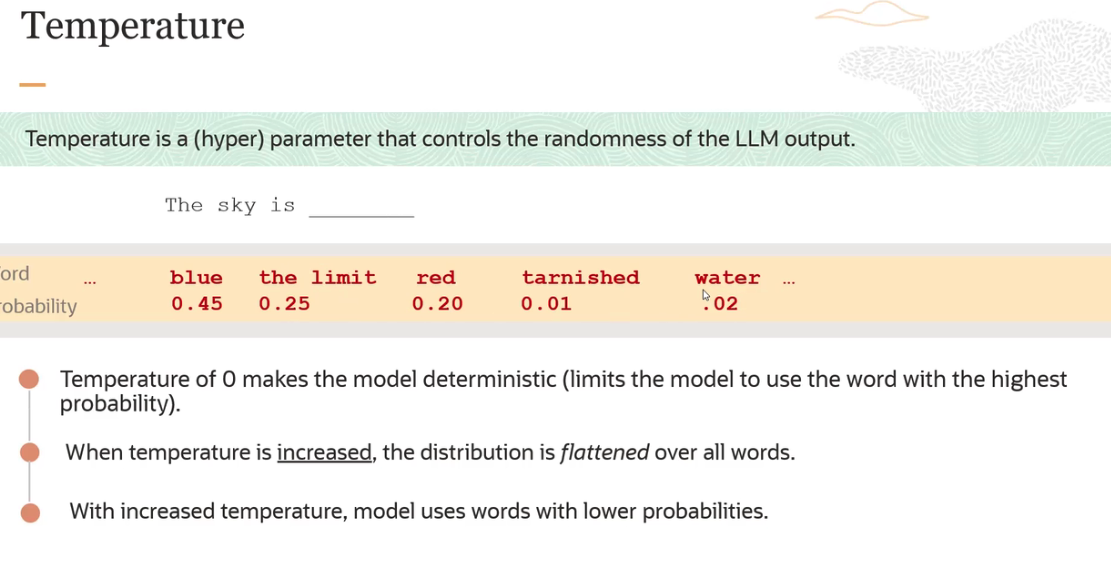
**More Details about parameters:**

The first parameter which is very important is temperature. Temperature is a hyperparameter that controls the randomness of the LLM output.

- The temperature setting basically tells it which of these words it can use as the next word.

- If you set the temperature to zero, it makes the model deterministic.

- Means it limits the model to use the word with the highest probability. So in this case, blue.

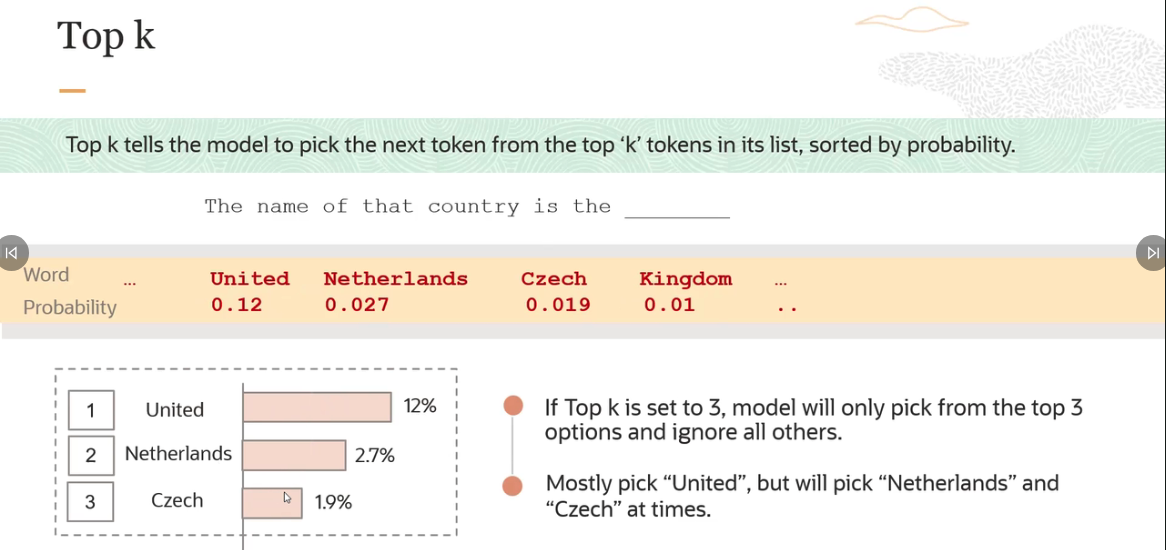


- If you increase the temperature, the distribution is flattened over all words. Means the model uses words with lower probability. So it could pick the next word here, which is the limit, which has a lower probability or it could even pick something like water. This is when you say the model has become more creative versus more deterministic.

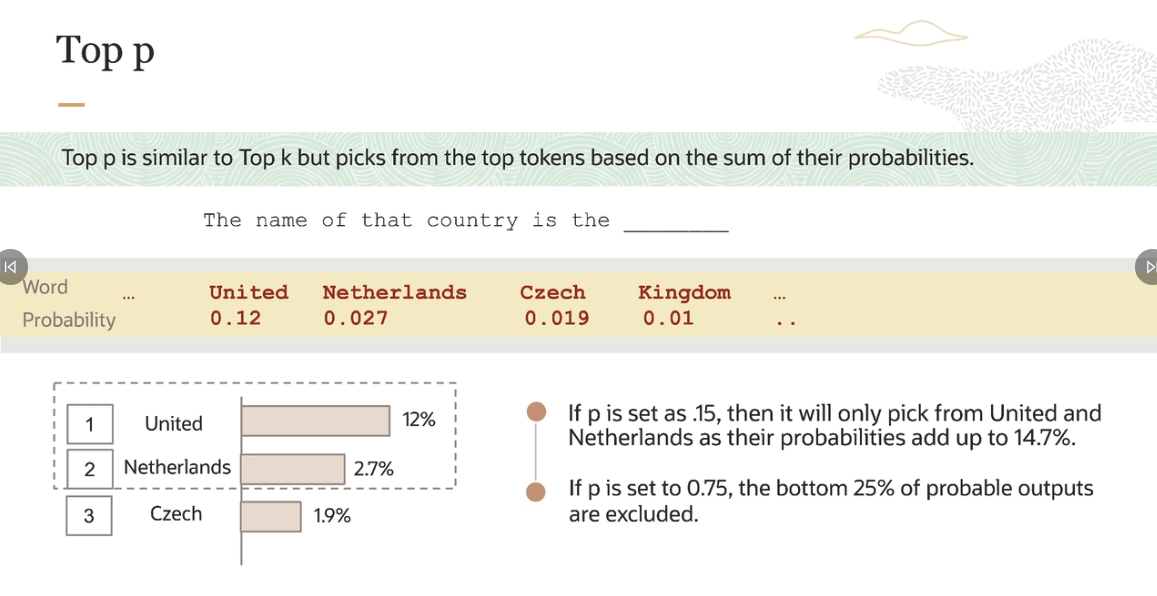
**Top k:** Top k basically tells the model to pick the next token from the top k tokens in its list sorted by probability.

- If you set your top k to three, basically, the model will only pick from the top three options. which are United, Netherlands, and Czech based on their probability and ignore all the other options.

- It will mostly pick United most of the time, but sometimes, it can also pick Netherlands and Czech.

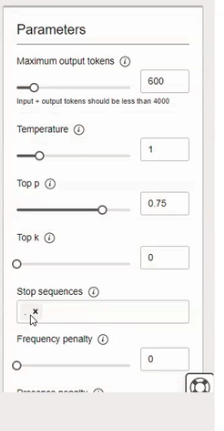
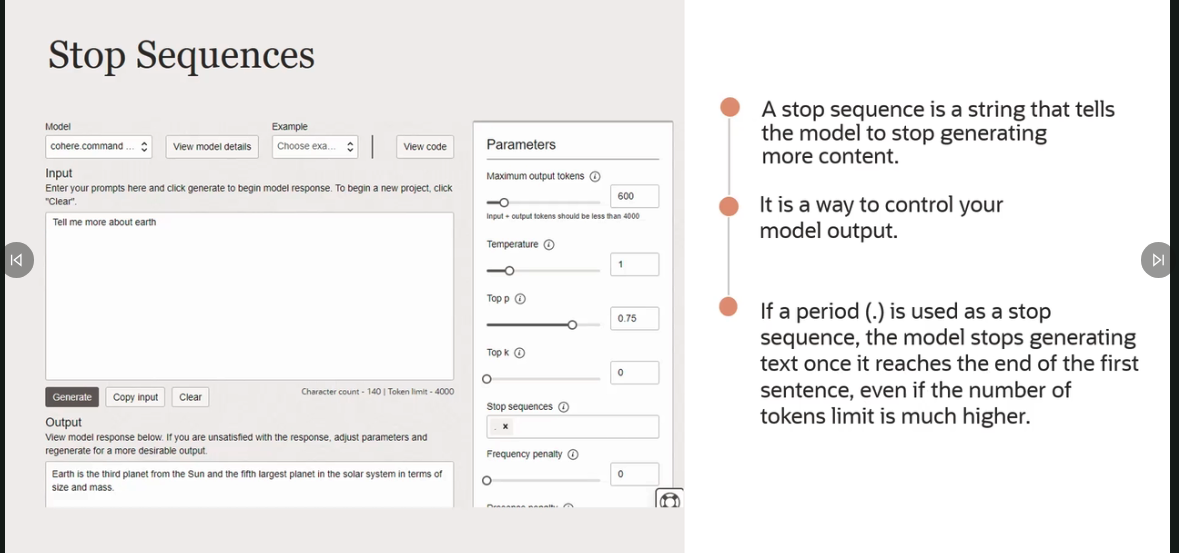


**Top p:** It is similar to top k but picks from the top tokens based on the sum of their probabilities.



**Stop Sequence**: I have set stop sequence as a period. And basically, stop sequence is a string that tells the model to stop generating more content. It's a way to control your model output.

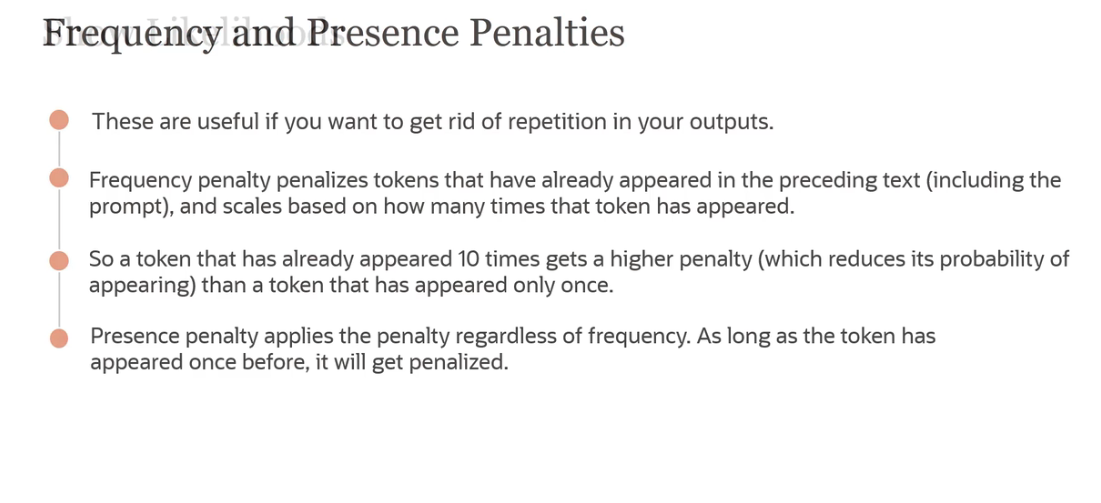
- For example, if a period is used as a stop sequence, the model stops generating text once it reaches the end of the first sentence, even if the number of tokens limit is much higher.

**Frequency and presence panalties:** The key thing to note here is these are useful if you want to get rid of repetitions in your output, you want your output to have less repetitive text.

**-** Frequency penalty penalizes tokens that have already appeared in the preceding text, including the prompt and scales based on how many times that token has appeared. So for example, if a token that has already appeared 10 times gets a higher penalty, which reduces its probability of appearing as a next token than a token that has only appeared only once. So it penalizes based on the frequency.

**-** Presence penalty applies the penalty regardless of frequency. As long as the token has appeared once before, it will get penalized. So the idea is you want your output to be less repetitive text.



**Show Likelihood:** Every time a new token is to be generated, a number between minus 15 and 0 is assigned to all the tokens. Tokens with higher numbers are more likely to follow the current token.

So for example, consider a phrase, like, this is my favorite dash. The next words which could follow, it's more likely that the word food or book is followed by the word favorite rather than zebra. So you can see here, book has a higher score than zebra. And food has a higher score than zebra. So the likelihood of book or food to follow favorite is much higher than the likelihood of zebra to follow favorite in this particular phrase. So this is how show likelihoods works.

