

LABORATORIO

Ajuste de los parámetros del modelo IBM Granite para perfeccionar el resultado

Contenido

Introducción	2
Requisitos de software	2
Objetivo	2
Pasos del laboratorio	2
Duración estimada para completarlo	3
Escenario	3
Antecedentes	3
Desafío	3
Solución	3
Paso 1: Confirma la configuración del software	4
Descripción general	4
Paso 2: Prueba el resultado de la clasificación inicial con valores predeterminados para los parámetros de la solicitud.....	5
Descripción general	5
Instrucciones	5
Paso 3: Ajusta un solo parámetro para mejorar el resultado de la clasificación.....	10
Descripción general	10
Instrucciones	10
Paso 4: Ajusta varios parámetros para refinar aún más el resultado de la clasificación	12
Descripción general	12
Instrucciones	12
Paso 5: Prueba el resultado del resumen inicial con valores predeterminados para los parámetros ..	18
Descripción general	18
Instrucciones	18
Paso 6: Ajusta un solo parámetro para mejorar el resultado del resumen	23
Descripción general	23
Instrucciones	23
Paso 7: Ajusta varios parámetros para mejorar el resultado del resumen	25
Descripción general	25
Instrucciones	25
Conclusión	28

Introducción

En este laboratorio, perfeccionarás el resultado del modelo IBM Granite ajustando parámetros clave para mejorar el rendimiento de las tareas de clasificación y resumen. Basándote en los flujos de trabajo automatizados implementados en el laboratorio anterior, ahora explorarás cómo el ajuste de parámetros específicos del modelo para la solicitud, como `max_tokens`, `top_k`, `top_p`, `repetition_penalty` y `stopping_sequence`, afecta la calidad de los resultados.

Estas mejoras te ayudarán a lograr resultados consistentes y prácticos adaptados a los requisitos específicos de las partes interesadas.

Requisitos de software

Para completar este laboratorio, debes tener acceso a lo siguiente:

- **Cuenta de Replicate:** Crea una cuenta de Replicate y obtén un token de API.
- **Google Colab:** Crea una cuenta de Google para utilizar Colab para ejecutar el cuaderno.
- **Paquetes Python:** Instala los paquetes necesarios (como `langchain_community`, sistema operativo, `replicate`, `google.colab`).

Objetivo

Después de completar este laboratorio, deberás ser capaz de:

- Ajustar los parámetros del modelo de IBM Granite para la solicitud para perfeccionar el resultado

Pasos del laboratorio

Este laboratorio requiere que completes los siguientes pasos:

- Paso 1: Confirma la configuración del software
- Paso 2: Prueba el resultado de la clasificación inicial con valores predeterminados para los parámetros de la solicitud
- Paso 3: Ajusta un solo parámetro para mejorar el resultado de la clasificación
- Paso 4: Ajusta varios parámetros para perfeccionar aún más el resultado de la clasificación
- Paso 5: Prueba el resultado del resumen inicial con valores predeterminados para los parámetros
- Paso 6: Ajusta un solo parámetro para mejorar el resultado del resumen
- Paso 7: Ajusta varios parámetros para mejorar el resultado del resumen

Duración estimada para completarlo

30 minutos

Escenario**Antecedentes**

Como jefe de producto de SmartTechX, una empresa de teléfonos inteligentes de tamaño mediano, tus responsabilidades incluyen analizar las reseñas de los clientes para identificar tendencias de opinión, etiquetar aspectos prioritarios y resumir reuniones de equipo multifuncionales para brindar a las partes interesadas información útil para la toma de decisiones.

En el laboratorio anterior, automatizaste correctamente estos flujos de trabajo utilizando el modelo **granite-3.0-8b-instruct**. Sin embargo, las partes interesadas han destacado problemas con la consistencia y precisión de los resultados. Algunos resultados de clasificación carecen de etiquetado de aspectos prioritarios, mientras que los resúmenes de reuniones a veces omiten detalles clave o no brindan una estructura clara. Para abordar estas dificultades, perfeccionarás el resultado del modelo ajustando los parámetros clave y garantizando que se alineen con las expectativas de las partes interesadas.

Desafío

A pesar del éxito de la automatización, las partes interesadas plantearon varios problemas con el resultado del modelo:

- **Tarea de clasificación:** Algunos resultados incluyen frases redundantes o inconsistentes. El resultado también adolece de un etiquetado incompleto de los aspectos prioritarios, lo que afecta la usabilidad.
- **Tarea de resumen:** Algunos resúmenes son demasiado genéricos y no captan las decisiones ni las acciones pendientes clave. Además, la estructura de los resultados debe alinearse mejor con las expectativas de las partes interesadas.

Solución

Creas que puedes perfeccionar tanto el resultado de la clasificación como el del resumen ajustando los parámetros del modelo **granite-3.0-8b-instruct** de las siguientes maneras:

- Usar **max_tokens** para controlar la longitud del resultado, garantizando respuestas concisas y específicas.
- Experimentar con parámetros como **top_k**, **top_p**, **repetition_penalty** y **stopping_sequence** para mejorar la coherencia, pertinencia y estructura del resultado.
- Evalúa el impacto de cada ajuste e itera según sea necesario para alinear los resultados con las necesidades de las partes interesadas.

Al ajustar estos parámetros, esperas perfeccionar el resultado tanto para las tareas de clasificación como de resumen, mejorando la eficacia, la precisión y la toma de decisiones en todos los departamentos.

Paso 1: Confirma la configuración del software

Descripción

En este paso, asegúrate de haber completado los siguientes pasos del laboratorio 1:

- Paso 1: Crea una cuenta de GitHub
- Paso 2: Configura tu entorno e inicializa el modelo de IBM Granite
- Paso 3: Regístrate en Google Colab

Si no has completado estos pasos, consulta el documento "Laboratorio 1: Solicitar a un modelo de IBM Granite

que clasifique y resuma datos" para obtener detalles y completar los pasos ahora.

Paso 2: Prueba el resultado de la clasificación inicial con valores predeterminados para los parámetros de la solicitud

Descripción

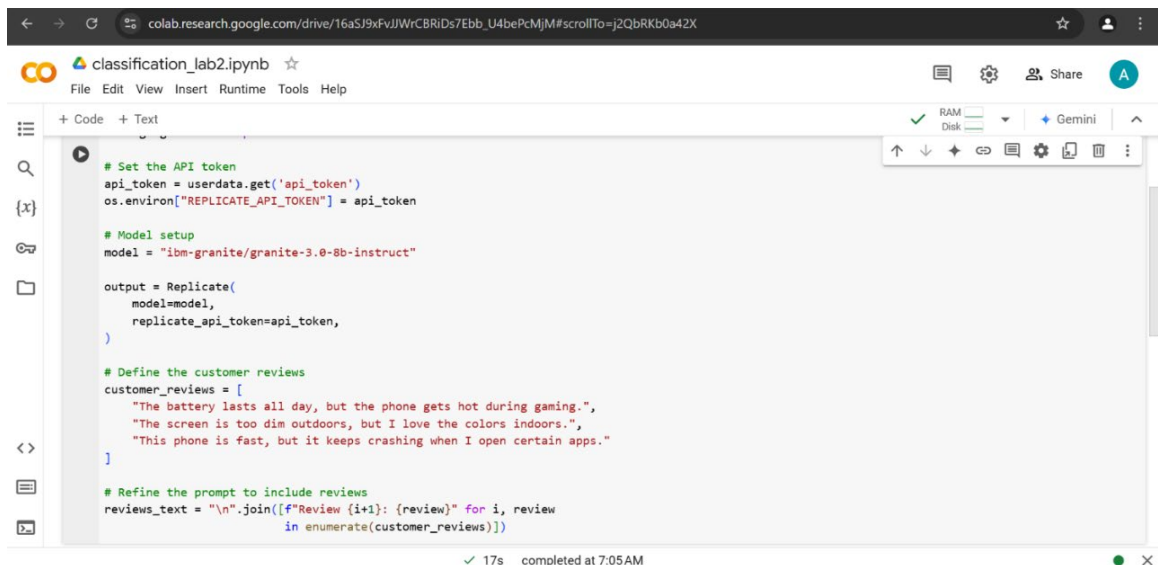
En este paso, ingresarás las reseñas de los clientes en el modelo y ejecutarás una solicitud básica con los parámetros predeterminados del modelo múltiple para clasificar las reseñas y analizar el resultado para identificar las áreas por mejorar.

Instrucciones

1. Ingresa muestras de reseñas de clientes para probar el modelo. Escribe estas reseñas de clientes en el campo **Instrucción**:

```
# Definir las reseñas de los clientes
customer_reviews = [
    "La batería dura todo el día, pero el teléfono se calienta mientras juego",
    "La pantalla es demasiado oscura en exteriores, pero me encantan los colores en interiores",
    "Este teléfono es rápido, pero falla cuando abro ciertas aplicaciones".
]
```

```
# Refinar la solicitud para incluir reseñas
reviews_text = "\n".join([f"Review {i+1}: {review}" for i, review in enumerate(customer_reviews)])
```



The screenshot shows a Google Colab notebook titled 'classification_lab2.ipynb'. The code in the notebook includes setting the API token, defining the model as 'ibm-granite/granite-3.0-8b-instruct', and defining a list of customer reviews. The reviews are: 'The battery lasts all day, but the phone gets hot during gaming.', 'The screen is too dim outdoors, but I love the colors indoors.', and 'This phone is fast, but it keeps crashing when I open certain apps.' The code also shows the reviews being joined into a single string with a newline separator. The notebook interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help), a toolbar with icons for code, text, and search, and a status bar at the bottom indicating '17s completed at 7:05 AM'.

```
# Set the API token
api_token = userdata.get('api_token')
os.environ["REPLICATE_API_TOKEN"] = api_token

# Model setup
model = "ibm-granite/granite-3.0-8b-instruct"

output = Replicate(
    model=model,
    replicate_api_token=api_token,
)

# Define the customer reviews
customer_reviews = [
    "The battery lasts all day, but the phone gets hot during gaming.",
    "The screen is too dim outdoors, but I love the colors indoors.",
    "This phone is fast, but it keeps crashing when I open certain apps."
]

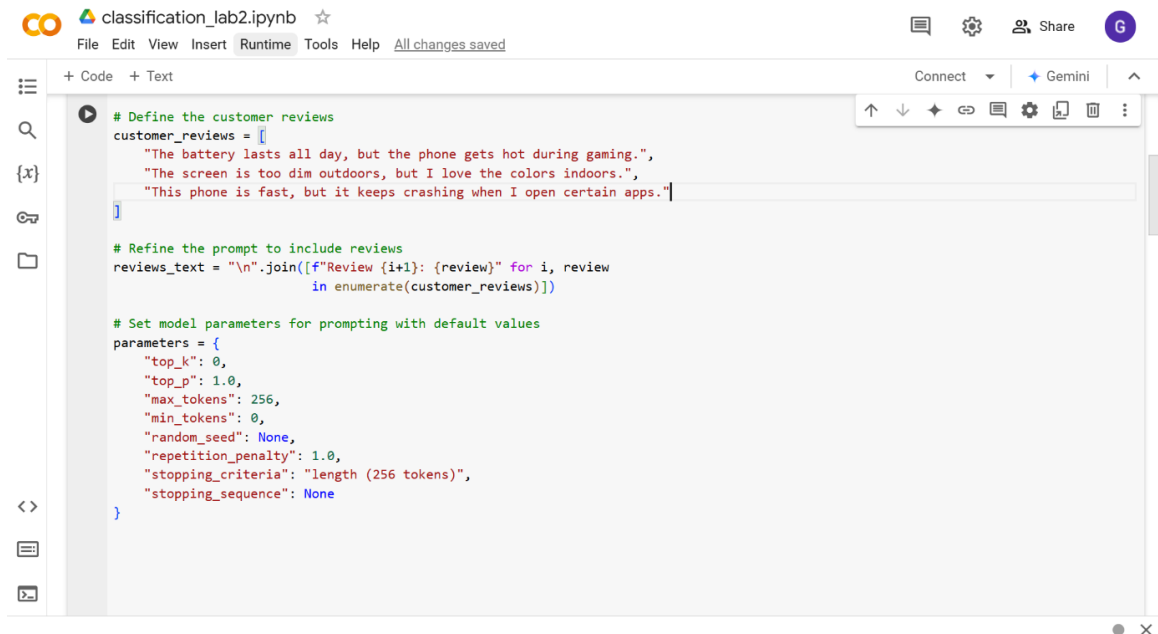
# Refine the prompt to include reviews
reviews_text = "\n".join([f"Review {i+1}: {review}" for i, review
                           in enumerate(customer_reviews)])
```

2. **Recuerda:** Ajustar los parámetros que controlan la selección, la longitud o la diversidad de los tokens puede mejorar significativamente la calidad de las respuestas. Estos incluyen:

- **top_k:** Limita la cantidad de posibles tokens siguientes considerados al generar texto, seleccionando entre las opciones "top k" más probables según su probabilidad. Esto enfoca el resultado y reduce la probabilidad de palabras inesperadas o irrelevantes, actuando como un filtro para las siguientes palabras más probables.
- **top_p:** Influye en la diversidad al restringir la selección de tokens a las opciones más probables dentro de un umbral de probabilidad acumulada. Garantiza que el modelo elija entre un conjunto más específico de opciones, mejorando la coherencia de la respuesta y manteniendo la diversidad.
- **max_tokens:** Controla la extensión máxima de la respuesta del modelo. Al establecer un límite de tokens, este parámetro evita respuestas demasiado largas y ayuda a centrar el resultado del modelo dentro de un tamaño manejable.
- **min_tokens:** Establece una longitud mínima para el resultado, lo que garantiza que el modelo genere respuestas suficientemente detalladas, lo que resulta útil para tareas que requieren explicaciones o resúmenes exhaustivos.
- **random_seed:** Garantiza la reproducibilidad al corregir la aleatoriedad en el resultado del modelo.
- **repetition_penalty:** Reduce la redundancia al penalizar los tokens repetidos en el resultado. Esto incentiva al modelo a generar un lenguaje más diverso y variado, evitando que repita las mismas palabras o frases innecesariamente.
- **stopping_criteria:** Define las condiciones bajo las cuales el modelo deja de generar tokens.
- **stopping_sequence:** Detiene la respuesta cuando el modelo encuentra una cadena predefinida, como un carácter de línea nueva o una palabra específica.

Introduce los valores predeterminados para los parámetros. Escribe este código en el campo Instrucción:

```
# Establecer los parámetros del modelo para la formulación de la
solicitud con valores predeterminados
parameters = {
    "top_k": 0,
    "top_p": 1.0,
    "max_tokens": 256,
    "min_tokens": 0,
    "random_seed": None,
    "repetition_penalty": 1.0,
    "stopping_criteria": "length (256 tokens)",
    "stopping_sequence": None
}
```



The screenshot shows a Jupyter Notebook titled 'classification_lab2.ipynb'. The code defines a list of customer reviews, refines the prompt to include these reviews, and sets model parameters for prompting with default values. The parameters include top_k, top_p, max_tokens, min_tokens, random_seed, repetition_penalty, stopping_criteria, and stopping_sequence.

```

# Define the customer reviews
customer_reviews = [
    "The battery lasts all day, but the phone gets hot during gaming.",
    "The screen is too dim outdoors, but I love the colors indoors.",
    "This phone is fast, but it keeps crashing when I open certain apps."
]

# Refine the prompt to include reviews
reviews_text = "\n".join([f"Review {i+1}: {review}" for i, review
                           in enumerate(customer_reviews)])

# Set model parameters for prompting with default values
parameters = {
    "top_k": 0,
    "top_p": 1.0,
    "max_tokens": 256,
    "min_tokens": 0,
    "random_seed": None,
    "repetition_penalty": 1.0,
    "stopping_criteria": "length (256 tokens)",
    "stopping_sequence": None
}

```

- Introduce la solicitud inicial. Escribe este código en el campo Instrucción:

```

# Agregar solicitud inicial
refined_prompt = f"""
Clasifica estas reseñas como positivas, negativas o mixtas y
etiqueta los aspectos prioritarios pertinentes, como la duración de
la batería, la calidad de la pantalla o el rendimiento

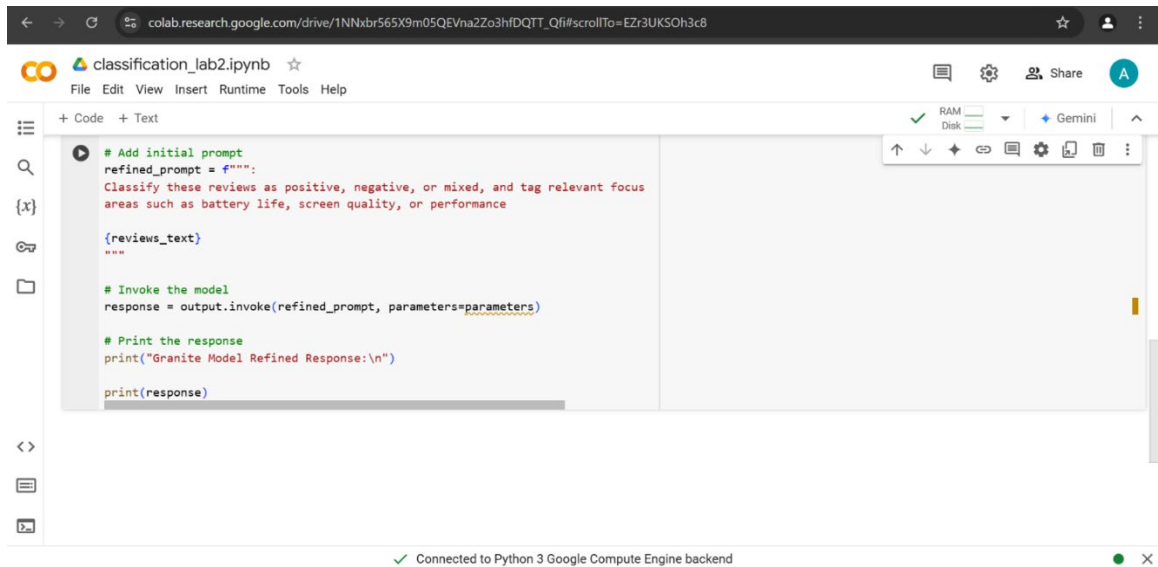
{reviews_text}
"""

# Invocar el modelo
respuesta = output.invoke(refined_prompt, parameters=parameters)

# Imprimir la respuesta
print("Granite Model Refined Response:\n")

print(response)

```

```
# Add initial prompt
refined_prompt = f"""
Classify these reviews as positive, negative, or mixed, and tag relevant focus
areas such as battery life, screen quality, or performance

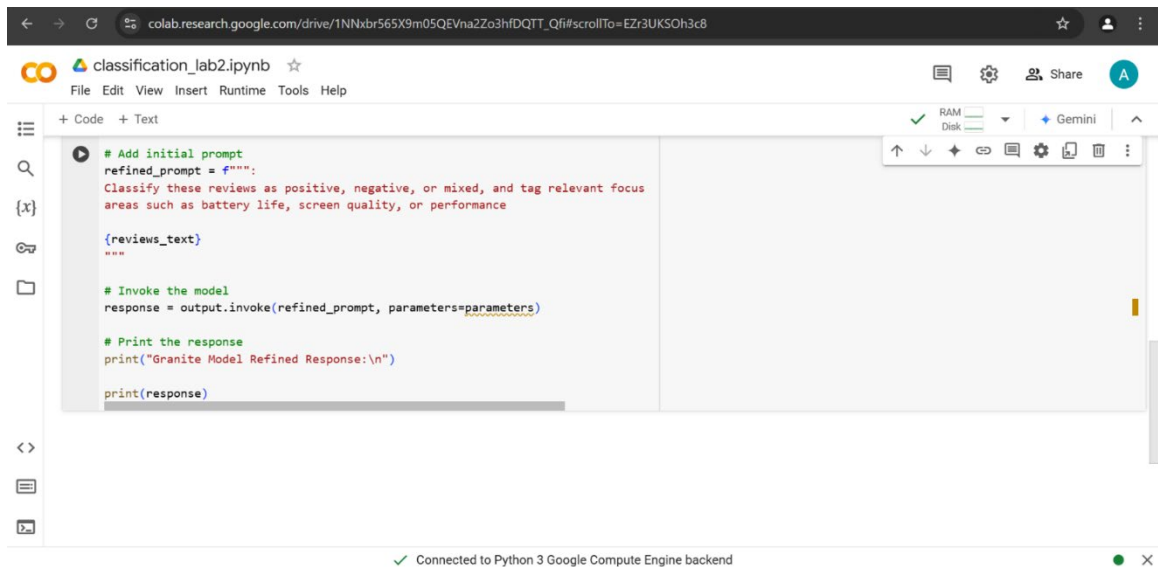
{reviews_text}
"""

# Invoke the model
response = output.invoke(refined_prompt, parameters=parameters)

# Print the response
print("Granite Model Refined Response:\n")
print(response)
```

Connected to Python 3 Google Compute Engine backend

4. Selecciona el botón **Play** para ejecutar la solicitud inicial para probar los valores predeterminados de los parámetros.



```
# Add initial prompt
refined_prompt = f"""
Classify these reviews as positive, negative, or mixed, and tag relevant focus
areas such as battery life, screen quality, or performance

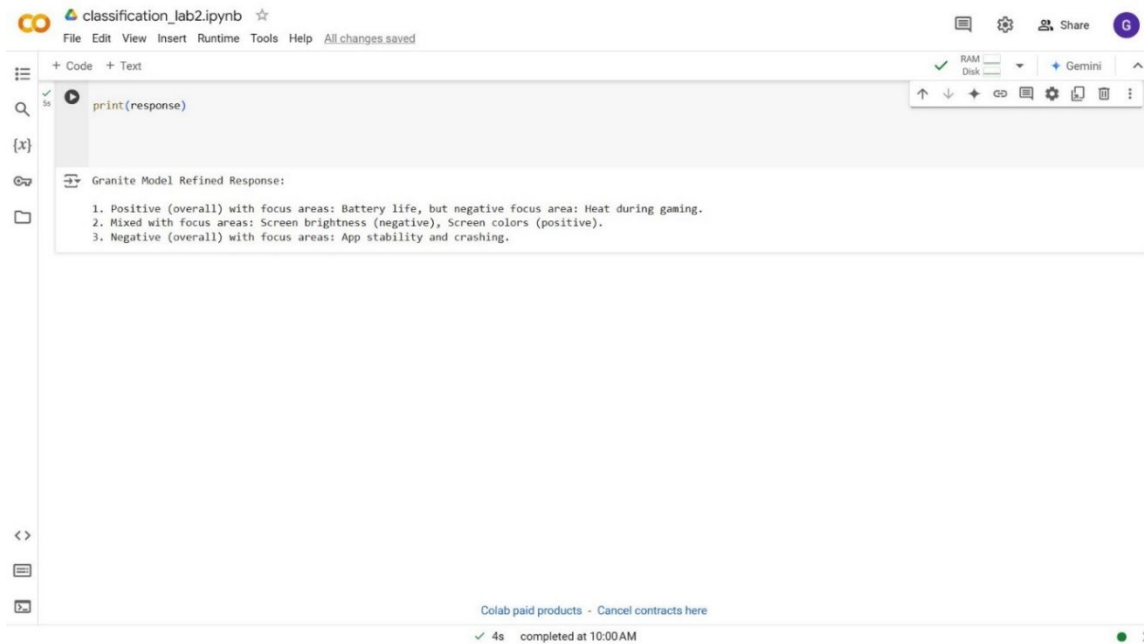
{reviews_text}
"""

# Invoke the model
response = output.invoke(refined_prompt, parameters=parameters)

# Print the response
print("Granite Model Refined Response:\n")
print(response)
```

Connected to Python 3 Google Compute Engine backend

5. Lee el resultado que genera el modelo. Observa que el resultado resalta los aspectos prioritarios de manera eficaz, abordando los aspectos positivos y negativos. Sin embargo, la respuesta no es concisa, por lo que es menos adecuada para una evaluación rápida. Para solucionar esto, es necesario mejorar la solicitud ajustando el parámetro `top_k` para garantizar una categorización de opiniones más granular y concisa.



The screenshot shows a Jupyter Notebook titled 'classification_lab2.ipynb'. The code cell contains the line `print(response)`. The output cell displays the 'Granite Model Refined Response' as a numbered list:

```
Granite Model Refined Response:
```

1. Positive (overall) with focus areas: Battery life, but negative focus area: Heat during gaming.
2. Mixed with focus areas: Screen brightness (negative), Screen colors (positive).
3. Negative (overall) with focus areas: App stability and crashing.

The bottom status bar indicates the execution was completed at 10:00 AM.

Paso 3: Ajusta un solo parámetro para mejorar el resultado de la clasificación

Descripción

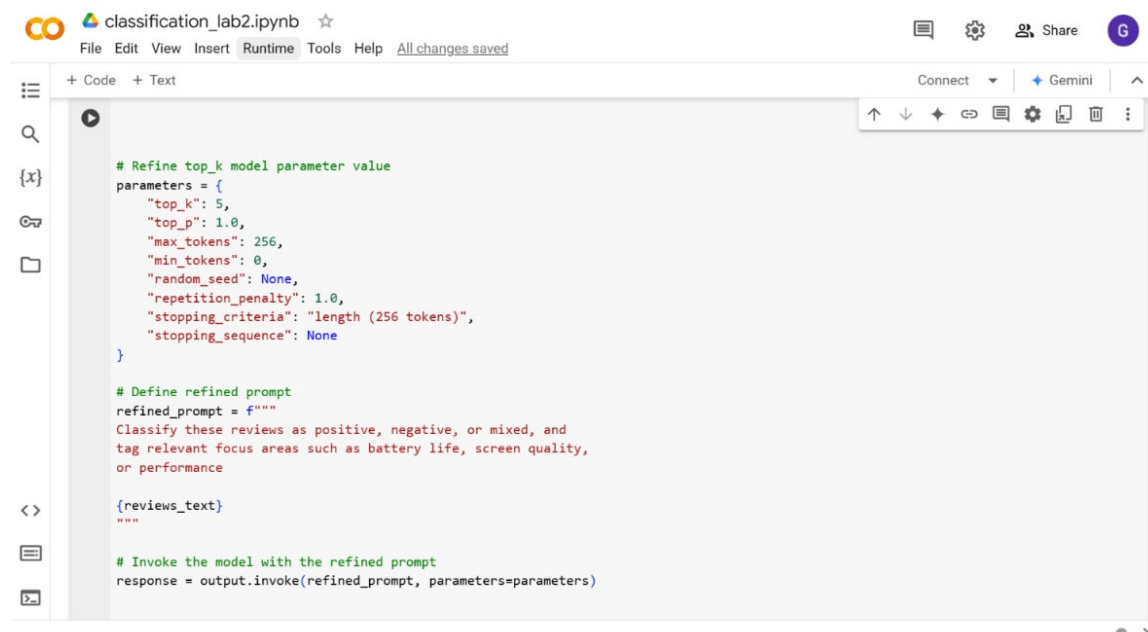
En este paso, ajustarás el parámetro `top_k` para agregar granularidad al categorizar explícitamente cada opinión de una reseña.

Instrucciones

1. **Recuerda:** Ajustar el parámetro `top_k` ayuda al modelo a centrarse en las opciones más probables, mejorando la precisión en sus clasificaciones. Al limitar el valor `top_k` a 5, el modelo considera solo los 5 tokens más probables, lo que reduce la ambigüedad y mejora la precisión del resultado.

Actualiza el parámetro `top_k` configurándolo en 5. Escribe este código en el campo **Instrucción:**

```
# Refinar el valor del parámetro top_k del modelo
parameters = {
    "top_k": 5,
    "top_p": 1.0,
    "max_tokens": 256,
    "min_tokens": 0,
    "random_seed": None,
    "repetition_penalty": 1.0,
    "stopping_criteria": "length (256 tokens)",
    "stopping_sequence": None
}
```



The screenshot shows a Jupyter Notebook titled 'classification_lab2.ipynb'. The code is written in a cell and includes comments in green. The code defines a dictionary 'parameters' with the following values: 'top_k': 5, 'top_p': 1.0, 'max_tokens': 256, 'min_tokens': 0, 'random_seed': None, 'repetition_penalty': 1.0, 'stopping_criteria': 'length (256 tokens)', and 'stopping_sequence': None. Below this, a prompt is defined: 'Classify these reviews as positive, negative, or mixed, and tag relevant focus areas such as battery life, screen quality, or performance'. The prompt is enclosed in triple quotes and labeled as 'reviews_text'. Finally, the code invokes the model using 'output.invoke(refined_prompt, parameters=parameters)'.

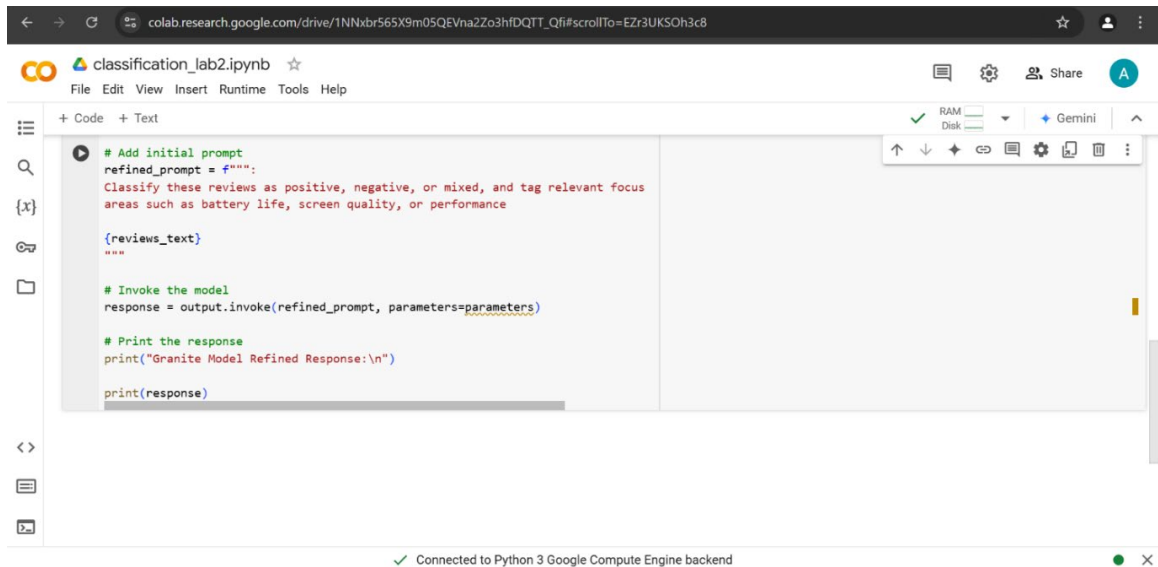
```
# Refine top_k model parameter value
parameters = {
    "top_k": 5,
    "top_p": 1.0,
    "max_tokens": 256,
    "min_tokens": 0,
    "random_seed": None,
    "repetition_penalty": 1.0,
    "stopping_criteria": "length (256 tokens)",
    "stopping_sequence": None
}

# Define refined prompt
refined_prompt = f"""
Classify these reviews as positive, negative, or mixed, and
tag relevant focus areas such as battery life, screen quality,
or performance

{reviews_text}
"""

# Invoke the model with the refined prompt
response = output.invoke(refined_prompt, parameters=parameters)
```

2. Selecciona el botón **Play** para ejecutar la solicitud.



```
# Add initial prompt
refined_prompt = f"""
Classify these reviews as positive, negative, or mixed, and tag relevant focus
areas such as battery life, screen quality, or performance

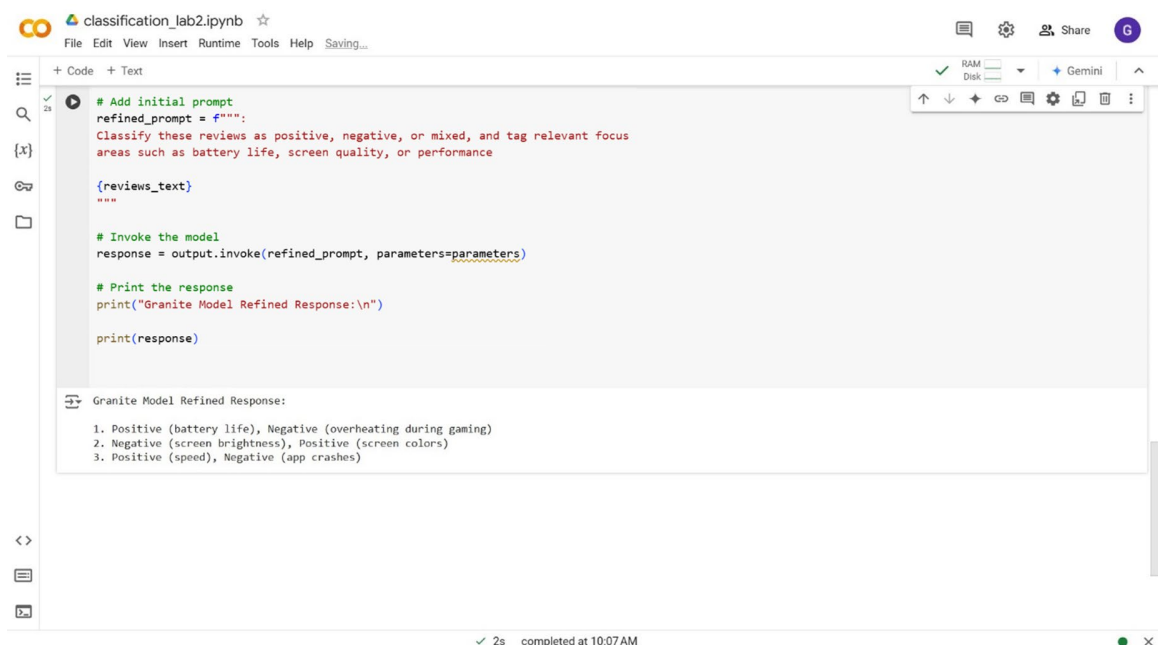
{reviews_text}
"""

# Invoke the model
response = output.invoke(refined_prompt, parameters=parameters)

# Print the response
print("Granite Model Refined Response:\n")
print(response)
```

3. Lee el resultado que genera el modelo. Ten en cuenta que el resultado categoriza eficazmente los sentimientos, pero podría ser más conciso. Si bien `top_k = 0` se centró en categorías amplias de sentimientos y proporcionó una clasificación de alto nivel, ajustar el parámetro a `top_k = 5` agregó granularidades al categorizar explícitamente cada sentimiento de una reseña.

Para mejorar aún más la respuesta, considera la posibilidad de ajustar múltiples parámetros en el siguiente paso para refinar la coherencia, precisión y pertinencia del resultado de la clasificación.



```
# Add initial prompt
refined_prompt = f"""
Classify these reviews as positive, negative, or mixed, and tag relevant focus
areas such as battery life, screen quality, or performance

{reviews_text}
"""

# Invoke the model
response = output.invoke(refined_prompt, parameters=parameters)

# Print the response
print("Granite Model Refined Response:\n")
print(response)
```

Granite Model Refined Response:

1. Positive (battery life), Negative (overheating during gaming)
2. Negative (screen brightness), Positive (screen colors)
3. Positive (speed), Negative (app crashes)

Paso 4: Ajusta varios parámetros para perfeccionar aún más el resultado de la clasificación

Descripción

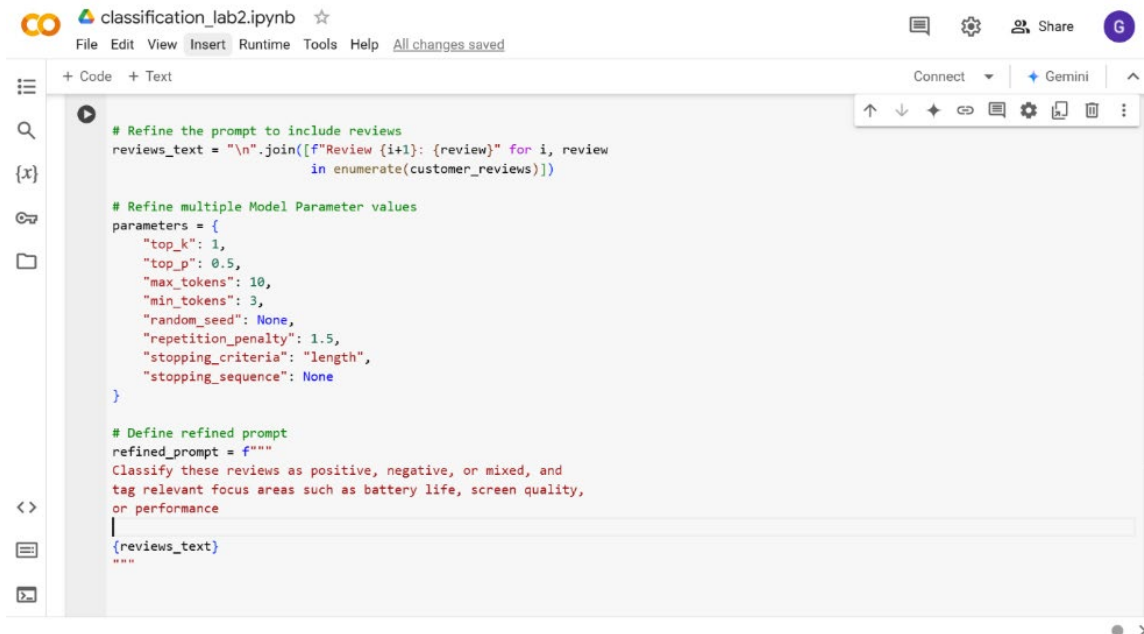
En este paso, ajustarás varios parámetros para perfeccionar la coherencia, precisión y pertinencia de los resultados de la clasificación.

Instrucciones

1. Ajusta los siguientes parámetros del modelo para mantener el resultado breve y conciso:
 - **top_k:** Ajustar el valor a 1 para restringir la respuesta al token más probable.
 - **top_p:** Ajustar el valor a 0.5 para filtrar aún más las respuestas en función de su coherencia y pertinencia.
 - **max_tokens:** Ajustar el valor a 10 para garantizar un resultado conciso.
 - **repetition_penalty:** Aumentar la penalización por repetición a 1.5 para evitar frases redundantes.

Introduce los valores de parámetros múltiples refinados. Escribe este código en el campo Instrucción:

```
# Refinar múltiples valores de parámetros del modelo
parameters = {
    "top_k": 1,
    "top_p": 0.5,
    "max_tokens": 10,
    "min_tokens": 3,
    "random_seed": None,
    "repetition_penalty": 1.5,
    "stopping_criteria": "length",
    "stopping_sequence": None
}
```



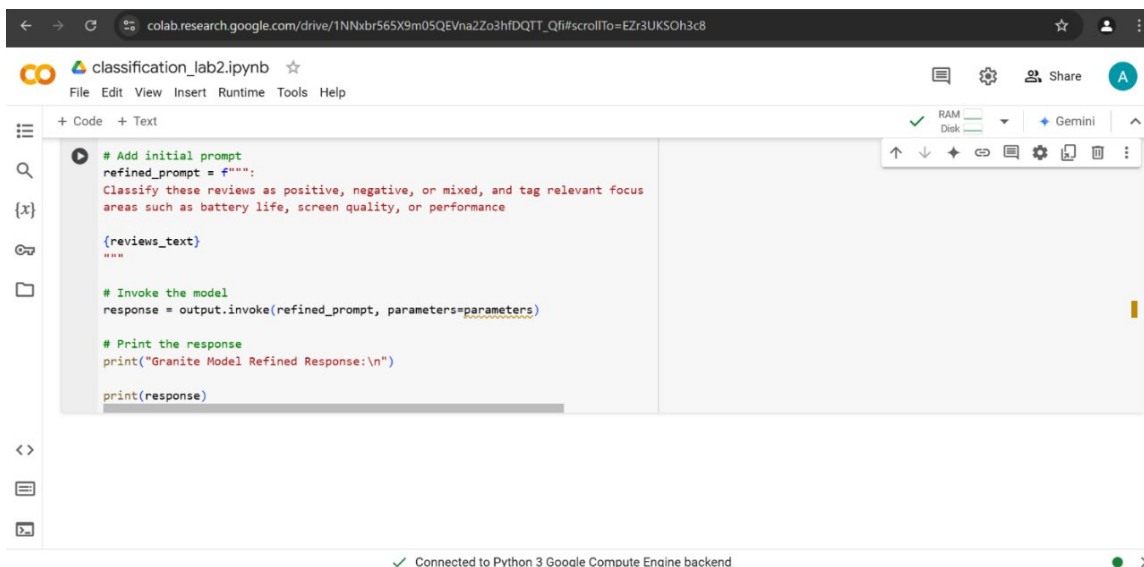
The screenshot shows a Jupyter Notebook titled 'classification_lab2.ipynb'. The code defines a function to refine a prompt by including reviews and sets various model parameters. The parameters include top_k, top_p, max_tokens, min_tokens, random_seed, repetition_penalty, stopping_criteria, and stopping_sequence. The refined prompt is then defined as a string that asks to classify reviews and tag relevant focus areas.

```
# Refine the prompt to include reviews
reviews_text = "\n".join([f"Review {i+1}: {review}" for i, review
                           in enumerate(customer_reviews)])

# Refine multiple Model Parameter values
parameters = {
    "top_k": 1,
    "top_p": 0.5,
    "max_tokens": 10,
    "min_tokens": 3,
    "random_seed": None,
    "repetition_penalty": 1.5,
    "stopping_criteria": "length",
    "stopping_sequence": None
}

# Define refined prompt
refined_prompt = f"""
Classify these reviews as positive, negative, or mixed, and
tag relevant focus areas such as battery life, screen quality,
or performance
{reviews_text}
"""
```

2. Selecciona el botón **Play** para ejecutar la solicitud.



The screenshot shows the same Jupyter Notebook with additional code to invoke the model and print the response. The code uses the 'output.invoke' method to call the model with the refined prompt and parameters, and then prints the resulting response.

```
# Add initial prompt
refined_prompt = f"""
Classify these reviews as positive, negative, or mixed, and tag relevant focus
areas such as battery life, screen quality, or performance
{reviews_text}
"""

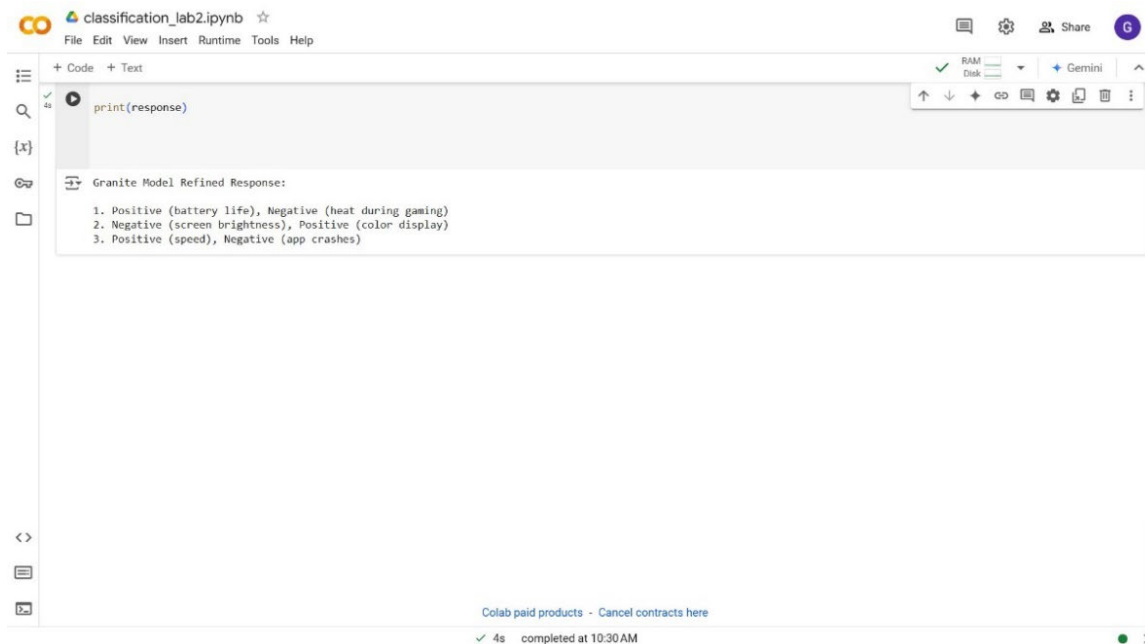
# Invoke the model
response = output.invoke(refined_prompt, parameters=parameters)

# Print the response
print("Granite Model Refined Response:\n")
print(response)
```

At the bottom of the notebook, a status bar indicates 'Connected to Python 3 Google Compute Engine backend'.

3. Lee el resultado que genera el modelo. Ten en cuenta que la respuesta categoriza eficazmente los sentimientos; aun así, podría beneficiarse de una mayor concisión. Ajustar `max_tokens = 10` garantizó un resultado conciso, mientras que `repetition_penalty = 1.5` evitó la redacción redundante. `top_k = 1` restringió la respuesta al token más probable, mientras que `top_p = 0.5` filtró aún más las respuestas en busca de coherencia y pertinencia. Sin embargo, aún hay margen para hacer que el resultado sea más conciso.

Ahora que has perfeccionado los parámetros iniciales, ajústalos nuevamente para que el resultado sea aún más conciso.



The screenshot shows a Jupyter Notebook titled 'classification_lab2.ipynb'. The code cell contains the line `print(response)`. The output cell displays the 'Granite Model Refined Response' as a numbered list of three items, each with positive and negative sentiment pairs in parentheses. The status bar at the bottom indicates 'Colab paid products - Cancel contracts here' and '4s completed at 10:30 AM'.

```
classification_lab2.ipynb
File Edit View Insert Runtime Tools Help

+ Code + Text
print(response)

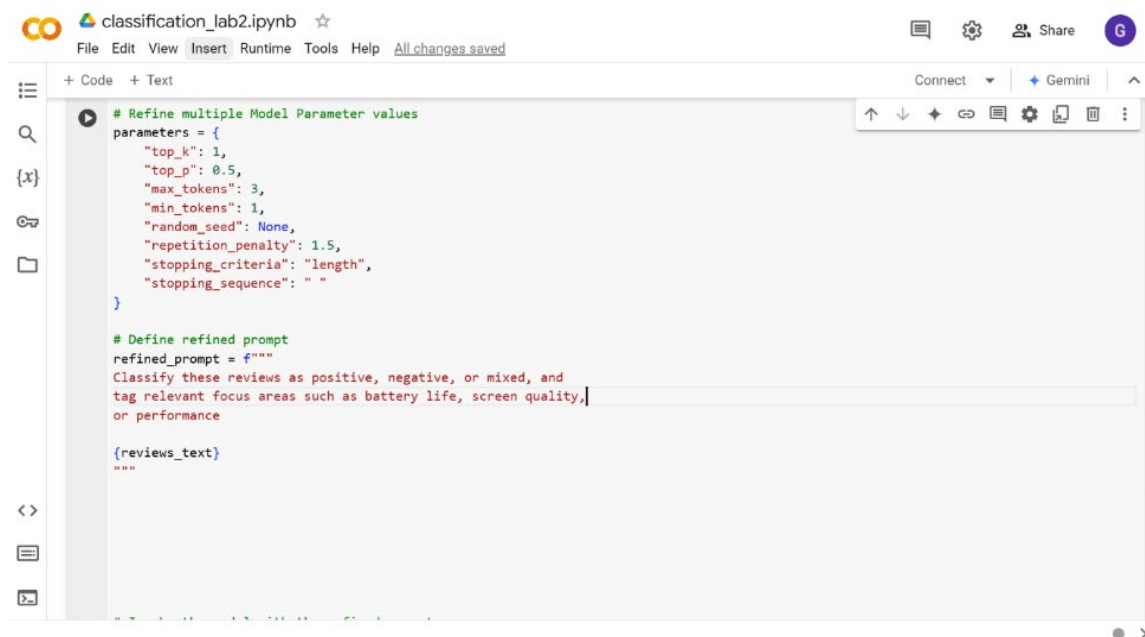
Granite Model Refined Response:
1. Positive (battery life), Negative (heat during gaming)
2. Negative (screen brightness), Positive (color display)
3. Positive (speed), Negative (app crashes)

Colab paid products - Cancel contracts here
4s completed at 10:30 AM
```

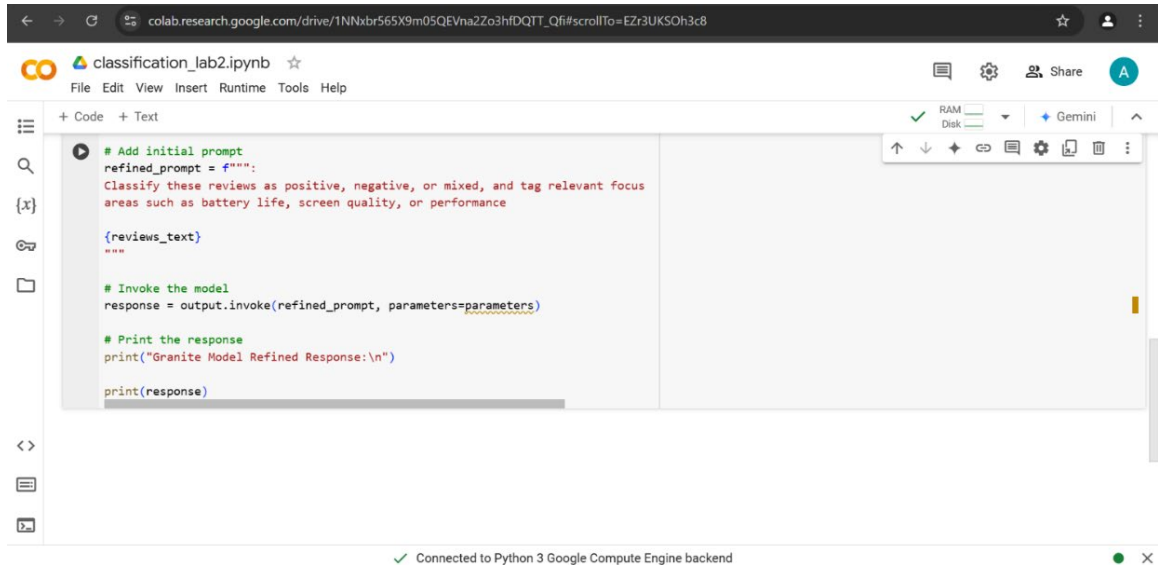
4. Ajusta nuevamente los siguientes parámetros para que el resultado sea aún más conciso:
- **max_tokens:** Redúcelo aún más a 3 para reforzar la brevedad.
 - **top_k:** Mantenlo en 1 para seleccionar el token más probable para el resultado determinista.
 - **top_p:** Mantenlo en 0.5 para centrarte en tokens de alta probabilidad.
 - **min_tokens:** Establécelo en 1 para garantizar una extensión mínima del resultado, lo que asegura que el modelo genere respuestas detalladas.
 - **random_seed:** Establécelo en **None** para un resultado variado o un valor fijo para la reproducibilidad.
 - **repetition_penalty:** Establécelo en 1.5 para penalizar los tokens repetidos y promover la variedad en la respuesta.
 - **stopping_criteria:** Establécelo en "length" para detener el modelo después de alcanzar un recuento de tokens especificado.
 - **stopping_sequence:** Establécelo como un espacio (" ") para forzar la finalización en pausas naturales, como los límites de palabras.

Introduce los valores de parámetros múltiples refinados. Escribe este código en el campo Instrucción:

```
#Refinar múltiples valores de parámetros del modelo
parameters = {
    "top_k": 1,
    "top_p": 0.5,
    "max_tokens": 3,
    "min_tokens": 1,
    "random_seed": None,
    "repetition_penalty": 1.5,
    "stopping_criteria": "length",
    "stopping_sequence": " "
}
```



5. Selecciona el botón **Play** para ejecutar la solicitud.



```
# Add initial prompt
refined_prompt = f"""
Classify these reviews as positive, negative, or mixed, and tag relevant focus
areas such as battery life, screen quality, or performance

{reviews_text}
"""

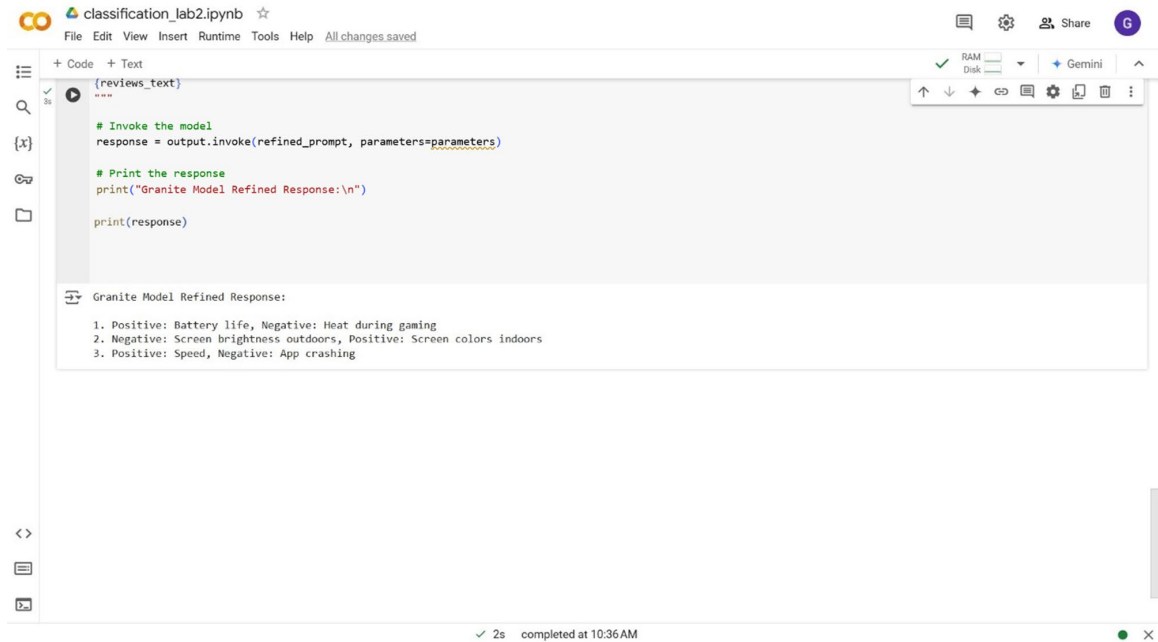
# Invoke the model
response = output.invoke(refined_prompt, parameters=parameters)

# Print the response
print("Granite Model Refined Response:\n")
print(response)
```

6. Lee el resultado que genera el modelo. Ten en cuenta que no se produjo ningún cambio en el resultado, incluso después de utilizar valores de token muy bajos y configurar la secuencia para que se detenga al encontrar un solo carácter de espacio.

Recuerda: Los valores de los parámetros del modelo tienen un umbral más allá del cual cualquier cambio en el valor no produciría ningún efecto en el resultado. Los modelos se entrenan para responder en una estructura determinada, en cuyo caso tanto los cambios de parámetros del modelo como las modificaciones de las solicitudes influyen en la respuesta de salida.

Has completado los ajustes de parámetros para obtener un resultado conciso. A continuación, probemos el resultado del resumen inicial con valores predeterminados para los parámetros ingresando la transcripción de la reunión.



The screenshot shows a Jupyter Notebook titled 'classification_lab2.ipynb'. The code cell contains the following Python code:

```
{reviews_text}  
.....  
  
# Invoke the model  
response = output.invoke(refined_prompt, parameters=parameters)  
  
# Print the response  
print("Granite Model Refined Response:\n")  
  
print(response)
```

The output cell displays the following text:

```
Granite Model Refined Response:  
1. Positive: Battery life, Negative: Heat during gaming  
2. Negative: Screen brightness outdoors, Positive: Screen colors indoors  
3. Positive: Speed, Negative: App crashing
```

The status bar at the bottom indicates '2s completed at 10:36AM'.

Paso 5: Prueba el resultado del resumen inicial con valores predeterminados para los parámetros

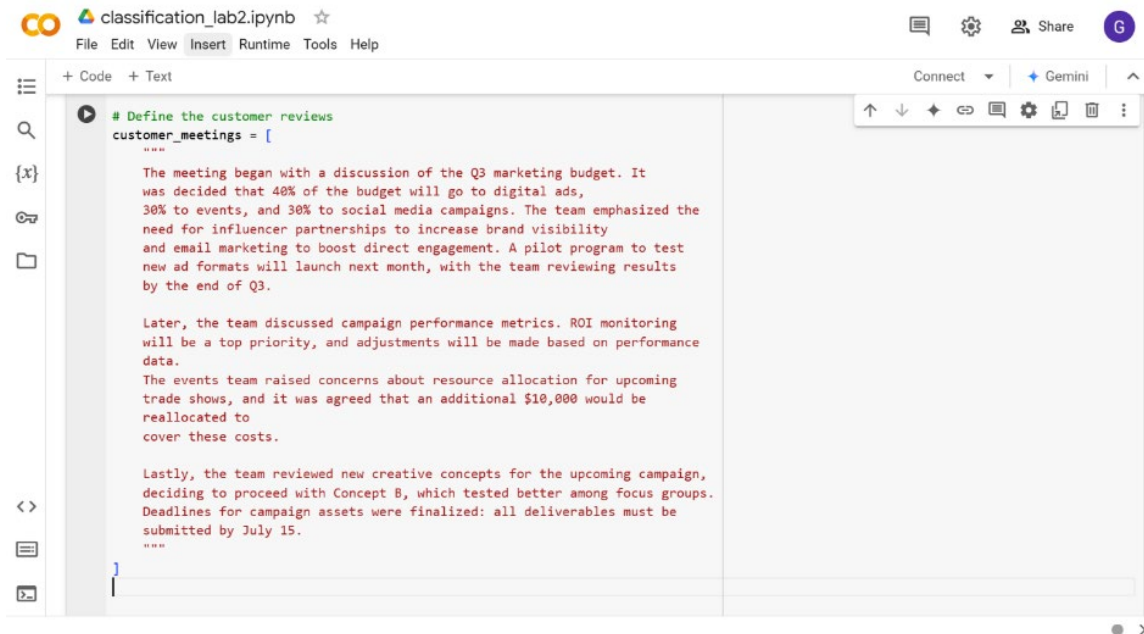
Descripción

En este paso, ingresarás una transcripción de la reunión en el modelo, ejecutarás una solicitud básica con los parámetros predeterminados para resumir la reunión y analizarás el resultado buscando posibilidades de mejorarlo.

Instrucciones

1. Ingresa la transcripción de la reunión simulando una reunión. Escribe esta transcripción en el campo Instrucción:

```
# Definir las reseñas de los clientes
customer_meetings = [
    "La reunión comenzó con una discusión sobre el presupuesto de marketing del tercer trimestre. Se decidió que el 40 % del presupuesto se destinará a anuncios digitales, el 30 % a eventos y el 30 % a campañas en redes sociales. El equipo enfatizó la necesidad de concretar asociaciones con influencers para aumentar la visibilidad de la marca y el marketing por correo electrónico para impulsar la interacción directa. El próximo mes se lanzará un programa piloto para probar nuevos formatos de anuncios y el equipo revisará los resultados a finales del tercer trimestre. Posteriormente, el equipo discutió las métricas de rendimiento de la campaña. El seguimiento del ROI será una máxima prioridad y se realizarán ajustes en función de los datos de rendimiento. El equipo de eventos planteó inquietudes sobre la asignación de recursos para las próximas ferias comerciales y se acordó que se reasignarían $10,000 adicionales para cubrir estos gastos. Por último, el equipo revisó nuevos conceptos creativos para la próxima campaña y decidió continuar con el Concepto B, que tuvo mejores resultados entre los grupos focales. Se ultimaron los plazos para la entrega de los activos de campaña: todos los entregables deben presentarse a más tardar el 15 de julio".
]
```



The screenshot shows a Jupyter Notebook titled 'classification_lab2.ipynb'. The code cell contains a function definition for 'customer_meetings' with three paragraphs of text. The first paragraph discusses the Q3 marketing budget allocation (40% to digital ads, 30% to events, 30% to social media) and the need for influencer partnerships. The second paragraph discusses campaign performance metrics, ROI monitoring, and resource allocation for upcoming trade shows. The third paragraph discusses creative concepts for the upcoming campaign and the finalization of campaign assets by July 15.

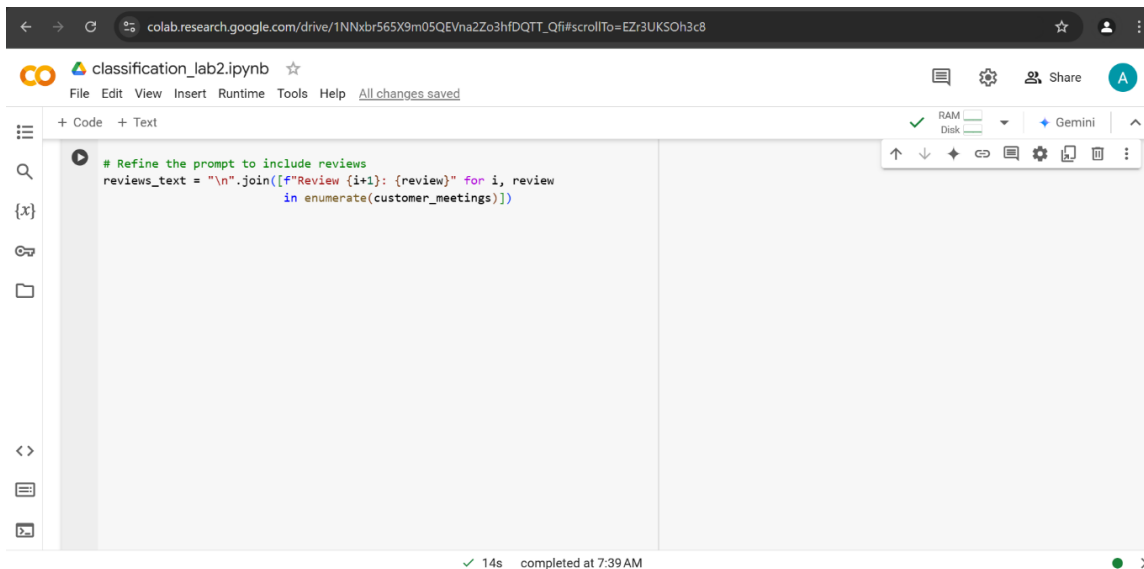
```
# Define the customer reviews
customer_meetings = [
    """
    The meeting began with a discussion of the Q3 marketing budget. It
    was decided that 40% of the budget will go to digital ads,
    30% to events, and 30% to social media campaigns. The team emphasized the
    need for influencer partnerships to increase brand visibility
    and email marketing to boost direct engagement. A pilot program to test
    new ad formats will launch next month, with the team reviewing results
    by the end of Q3.

    Later, the team discussed campaign performance metrics. ROI monitoring
    will be a top priority, and adjustments will be made based on performance
    data.
    The events team raised concerns about resource allocation for upcoming
    trade shows, and it was agreed that an additional $10,000 would be
    reallocated to
    cover these costs.

    Lastly, the team reviewed new creative concepts for the upcoming campaign,
    deciding to proceed with Concept B, which tested better among focus groups.
    Deadlines for campaign assets were finalized: all deliverables must be
    submitted by July 15.
    """
]
```

2. Ahora, ingresa la solicitud para incluir reseñas. Escribe esta solicitud en el campo Instrucción:

```
# Refinar la solicitud para incluir reseñas
reviews_text = "\n".join([f"Review {i+1}: {review}" for i,
review in enumerate(customer_meetings)])
```



The screenshot shows the same Jupyter Notebook interface, but the code cell now contains the refined prompt to include reviews. The code is identical to the previous one, but the function definition for 'customer_meetings' is not visible in this view. The status bar at the bottom indicates that the code was executed successfully, taking 14 seconds and completing at 7:39 AM.

```
# Refine the prompt to include reviews
reviews_text = "\n".join([f"Review {i+1}: {review}" for i, review
in enumerate(customer_meetings)])
```

✓ 14s completed at 7:39 AM

3. Ingresa los parámetros predeterminados y la solicitud mejorada para configurar la respuesta del modelo, centrándote en resumir los puntos clave, las decisiones y las acciones pendientes de la reunión. Escribe este código en el campo Instrucción:

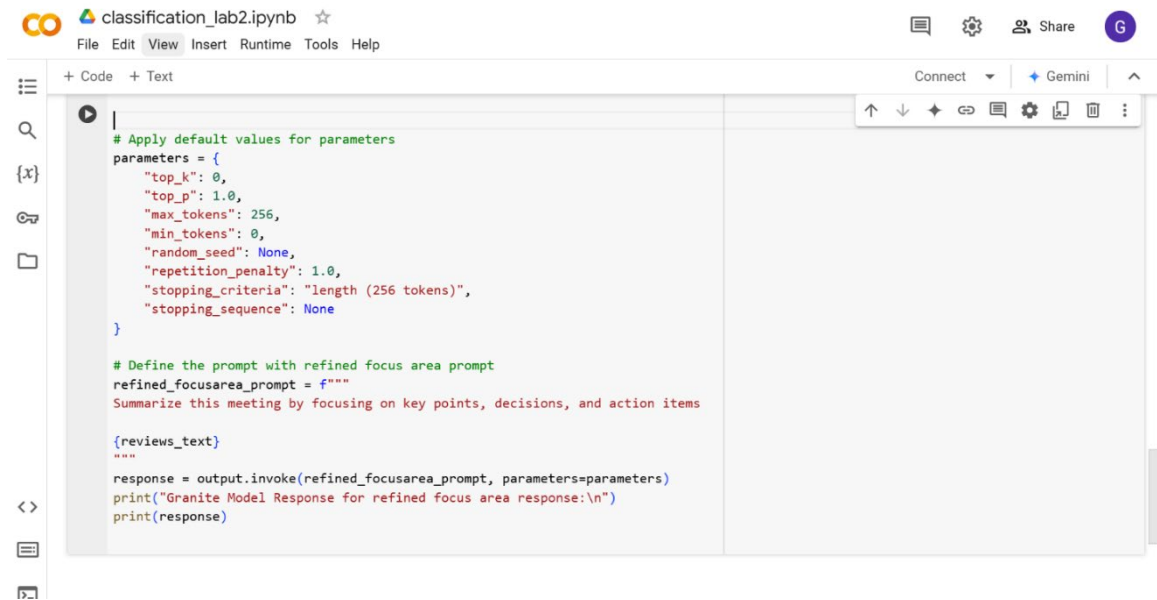
```
# Aplicar parámetros predeterminados
parameters = {
    "top_k": 0,
    "top_p": 1.0,
    "max_tokens": 256,
    "min_tokens": 0,
    "random_seed": None,
    "repetition_penalty": 1.0,
    "stopping_criteria": "length (256 tokens)",
    "stopping_sequence": None
}

# Solicitud refinada con ejemplo incluido
refined_focus_prompt = f"""
Resume esta reunión centrándote en los puntos clave, las
decisiones y las acciones pendientes.

{reviews_text}
"""

# Invocar el modelo con la solicitud de enfoque refinado
respuesta = output.invoke(refined_focus_prompt,
parameters=parameters)
# Imprimir la respuesta
print("Granite Model Response for refined focus area response:\n")

print(respuesta)
```



```
classification_lab2.ipynb
File Edit View Insert Runtime Tools Help

+ Code + Text

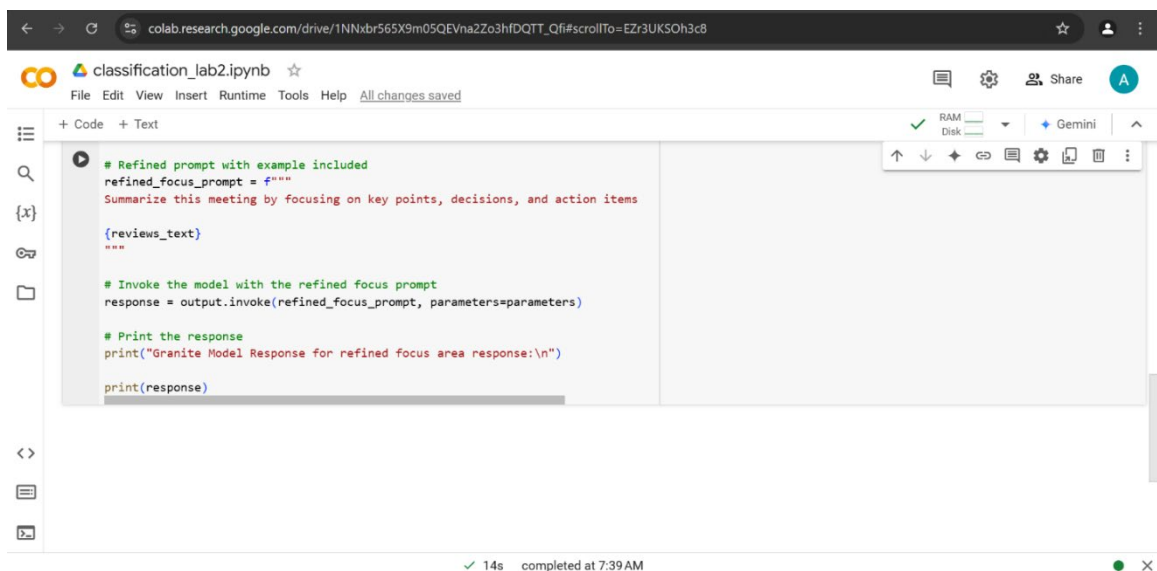
# Apply default values for parameters
parameters = {
    "top_k": 0,
    "top_p": 1.0,
    "max_tokens": 256,
    "min_tokens": 0,
    "random_seed": None,
    "repetition_penalty": 1.0,
    "stopping_criteria": "length (256 tokens)",
    "stopping_sequence": None
}

# Define the prompt with refined focus area prompt
refined_focusarea_prompt = f"""
Summarize this meeting by focusing on key points, decisions, and action items

{reviews_text}
"""

response = output.invoke(refined_focusarea_prompt, parameters=parameters)
print("Granite Model Response for refined focus area response:\n")
print(response)
```

4. Selecciona el botón **Play** para ejecutar la solicitud.



```
classification_lab2.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

# Refined prompt with example included
refined_focus_prompt = f"""
Summarize this meeting by focusing on key points, decisions, and action items

{reviews_text}
"""

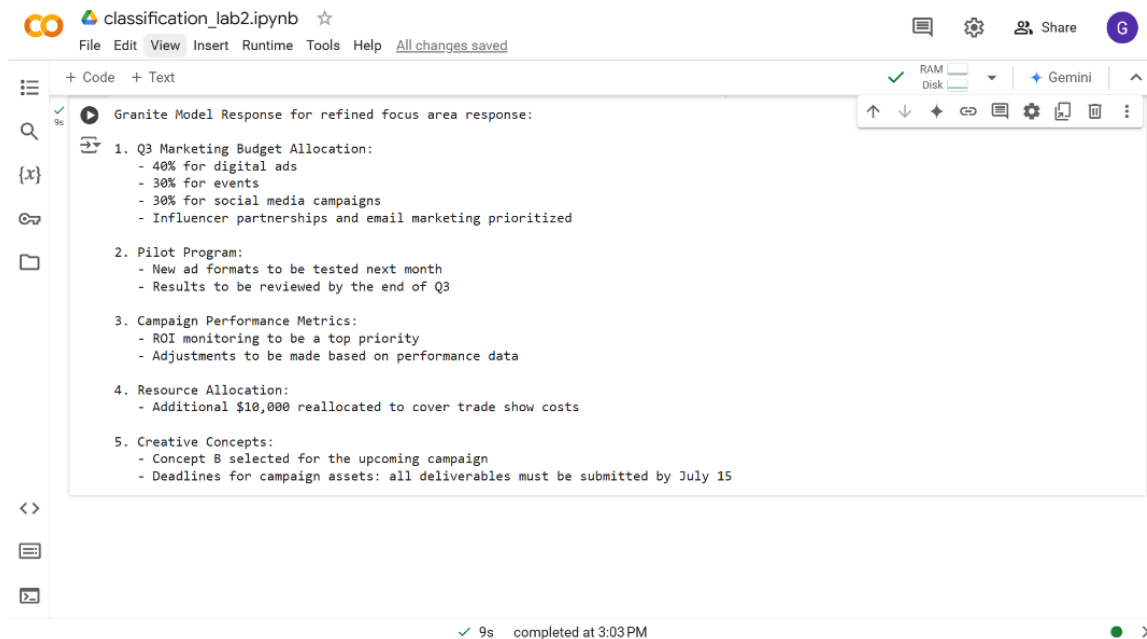
# Invoke the model with the refined focus prompt
response = output.invoke(refined_focus_prompt, parameters=parameters)

# Print the response
print("Granite Model Response for refined focus area response:\n")
print(response)
```

✓ 14s completed at 7:39 AM

5. Lee el resultado que genera el modelo. Ten en cuenta que el resultado proporciona un desglose detallado de las estrategias de marketing, pero podría ser demasiado detallado para escenarios que necesitan resúmenes más concisos. En muchos contextos, los encargados de la toma de decisiones o las partes interesadas prefieren resúmenes breves que destaquen los puntos clave, lo que significa un ahorro de tiempo y sin dejar de plasmar la esencia del contenido. Una respuesta más concisa es más efectiva, especialmente para presentaciones o informes ejecutivos.

Puedes mejorar el resultado ajustando el parámetro `max_tokens` en el siguiente paso.



The screenshot shows a Jupyter Notebook interface with a code cell. The code cell contains a single line of text: "Granite Model Response for refined focus area response:". Below this line, the output of the model is displayed as a numbered list with five items, each containing sub-points. The output is formatted with indentation for the sub-points. The notebook interface includes a top bar with the file name "classification_lab2.ipynb", a menu bar with options like File, Edit, View, Insert, Runtime, Tools, and Help, and a right sidebar with icons for RAM, Disk, and Gemini. The bottom status bar indicates "9s completed at 3:03 PM".

```
Granite Model Response for refined focus area response:
```

1. Q3 Marketing Budget Allocation:
 - 40% for digital ads
 - 30% for events
 - 30% for social media campaigns
 - Influencer partnerships and email marketing prioritized
2. Pilot Program:
 - New ad formats to be tested next month
 - Results to be reviewed by the end of Q3
3. Campaign Performance Metrics:
 - ROI monitoring to be a top priority
 - Adjustments to be made based on performance data
4. Resource Allocation:
 - Additional \$10,000 reallocated to cover trade show costs
5. Creative Concepts:
 - Concept B selected for the upcoming campaign
 - Deadlines for campaign assets: all deliverables must be submitted by July 15

9s completed at 3:03 PM

Paso 6: Ajusta un solo parámetro para mejorar el resultado del resumen

Descripción

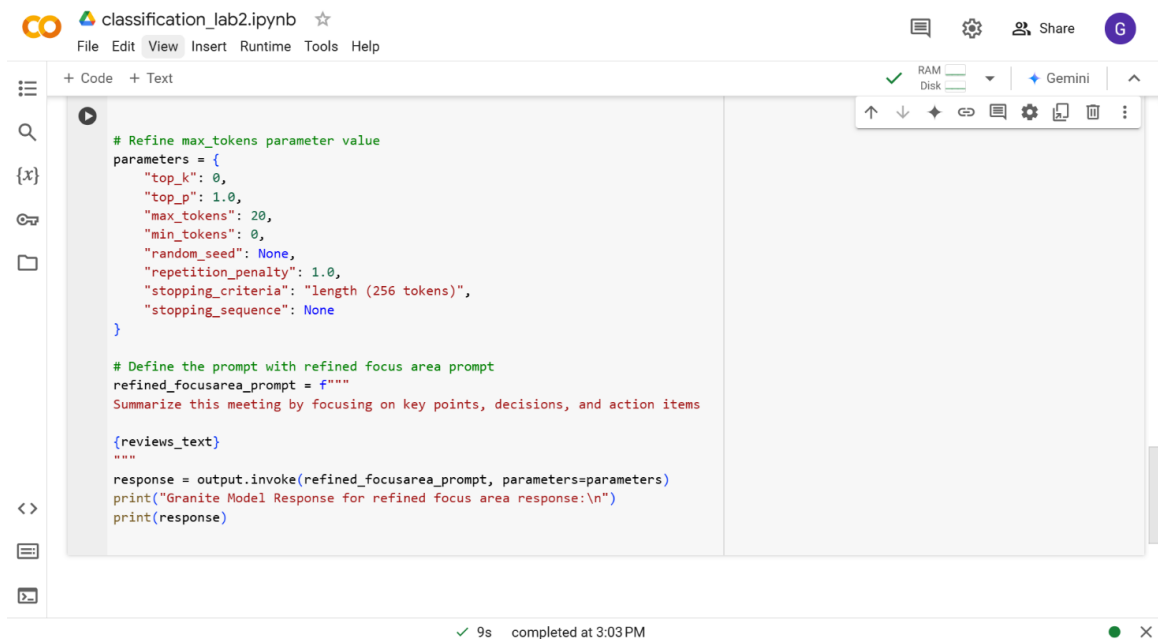
En este paso, ajustarás el parámetro `max_tokens` para permitir que el modelo genere resúmenes más concisos.

Instrucciones

1. Actualiza el parámetro `max_tokens` configurándolo en 20. Escribe este código en el campo **Instrucción**:

```
# Refinar el valor del parámetro max_tokens

parameters = {
    "top_k": 0,
    "top_p": 1.0,
    "max_tokens": 20,
    "min_tokens": 0,
    "random_seed": None,
    "repetition_penalty": 1.0,
    "stopping_criteria": "length (256 tokens)",
    "stopping_sequence": None
}
```



The screenshot shows a Jupyter Notebook titled 'classification_lab2.ipynb'. The code in the cell is as follows:

```
# Refine max_tokens parameter value
parameters = {
    "top_k": 0,
    "top_p": 1.0,
    "max_tokens": 20,
    "min_tokens": 0,
    "random_seed": None,
    "repetition_penalty": 1.0,
    "stopping_criteria": "length (256 tokens)",
    "stopping_sequence": None
}

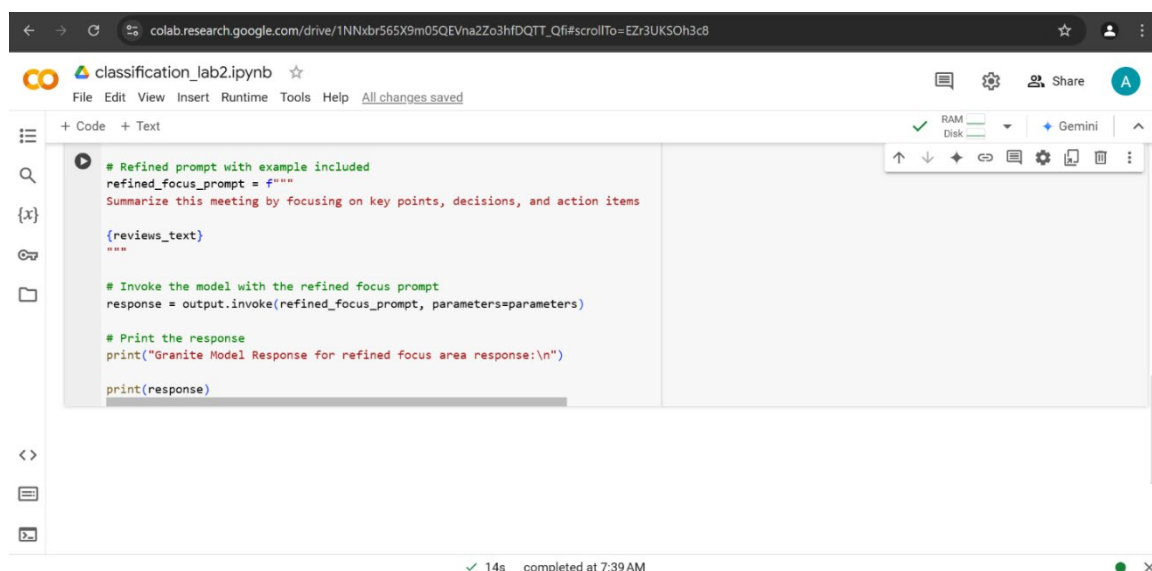
# Define the prompt with refined focus area prompt
refined_focusarea_prompt = f"""
Summarize this meeting by focusing on key points, decisions, and action items

{reviews_text}
"""

response = output.invoke(refined_focusarea_prompt, parameters=parameters)
print("Granite Model Response for refined focus area response:\n")
print(response)
```

The notebook interface includes a left sidebar with icons for file explorer, search, and other tools. The top bar shows the file name and various settings. The bottom status bar indicates the execution time as '9s' and 'completed at 3:03 PM'.

2. Selecciona el botón **Play** para ejecutar la solicitud.



```
# Refined prompt with example included
refined_focus_prompt = """
Summarize this meeting by focusing on key points, decisions, and action items

{reviews_text}
"""

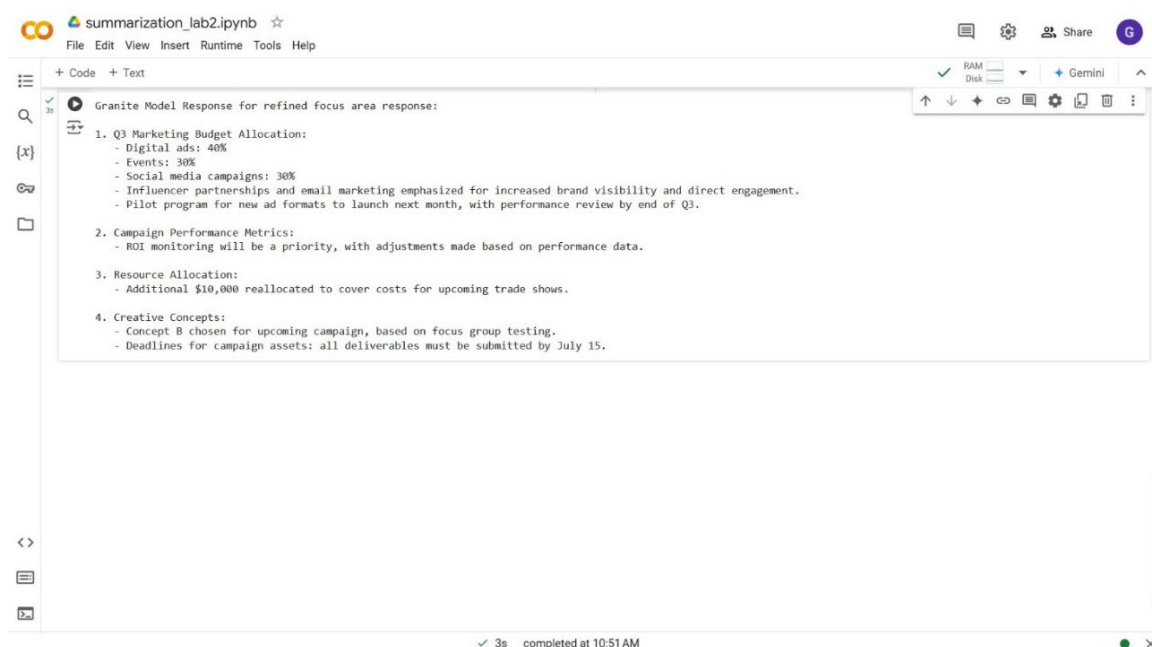
# Invoke the model with the refined focus prompt
response = output.invoke(refined_focus_prompt, parameters=parameters)

# Print the response
print("Granite Model Response for refined focus area response:\n")

print(response)
```

3. Lee el resultado que genera el modelo. Ten en cuenta que reducir el parámetro `max_tokens` a 20 (`max_tokens = 20`) condensó significativamente la respuesta, priorizando información esencial como la asignación de presupuesto, el enfoque de la campaña y las fechas límite. Este ajuste sintetizó detalles clave, eliminando información menos crítica en comparación con los valores de los parámetros predeterminados. Sin embargo, el resultado podría refinarse aún más evitando palabras repetitivas.

Puedes mejorar el resultado ajustando más parámetros en el siguiente paso.



```
Granite Model Response for refined focus area response:

1. Q3 Marketing Budget Allocation:
   - Digital ads: 40%
   - Events: 30%
   - Social media campaigns: 30%
   - Influencer partnerships and email marketing emphasized for increased brand visibility and direct engagement.
   - Pilot program for new ad formats to launch next month, with performance review by end of Q3.

2. Campaign Performance Metrics:
   - ROI monitoring will be a priority, with adjustments made based on performance data.

3. Resource Allocation:
   - Additional $10,000 reallocated to cover costs for upcoming trade shows.

4. Creative Concepts:
   - Concept B chosen for upcoming campaign, based on focus group testing.
   - Deadlines for campaign assets: all deliverables must be submitted by July 15.
```

Paso 7: Ajusta varios parámetros para mejorar el resultado del resumen

Descripción

En este paso, ajustarás varios parámetros para generar resúmenes bien estructurados.

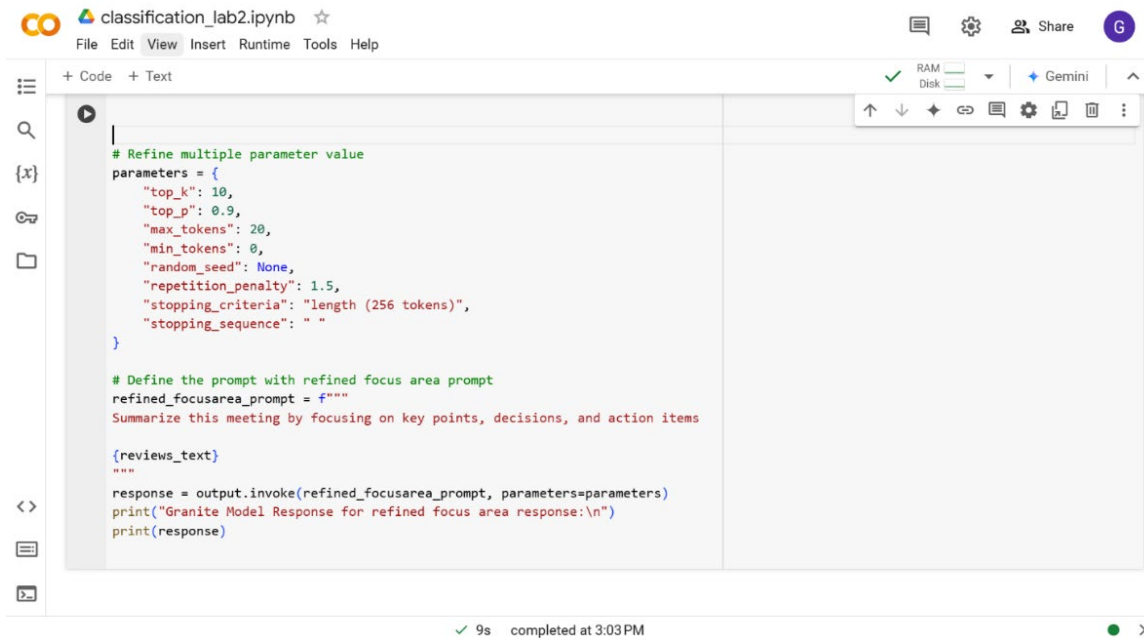
Instrucciones

1. Ajusta los siguientes parámetros para garantizar que mejore la calidad del resultado:
 - **top_p**: Mantenlo en 0.9 para obtener resultados específicos y de alta probabilidad.
 - **top_k**: Establéclo en 10 para permitir un poco más de variación en la selección de tokens manteniendo la coherencia.
 - **max_tokens**: Mantenlo en 20 para garantizar respuestas concisas.
 - **repetition_penalty**: Mantenlo en 1.5 para reducir la redundancia.
 - **stopping_sequence**: Mantenlo como un espacio (" ") para finalizar el resultado en las pausas naturales.

Introduce los valores refinados para los parámetros requeridos. Escribe este código en el campo Instrucción:

```
# Refinar múltiples valores de parámetros
```

```
parameters = {  
    "top_k": 10,  
    "top_p": 0.9,  
    "max_tokens": 20,  
    "min_tokens": 0,  
    "random_seed": None,  
    "repetition_penalty": 1.5,  
    "stopping_criteria": "length (256 tokens)",  
    "stopping_sequence": " "  
}
```



The screenshot shows a Jupyter Notebook titled 'classification_lab2.ipynb'. The code defines parameters for the model and a refined focus area prompt. The parameters include top_k, top_p, max_tokens, min_tokens, random_seed, repetition_penalty, stopping_criteria, and stopping_sequence. The refined focus area prompt is defined as a string that instructs the model to summarize a meeting by focusing on key points, decisions, and action items. The code then invokes the model with the refined focus area prompt and prints the response.

```
# Refine multiple parameter value
parameters = {
    "top_k": 10,
    "top_p": 0.9,
    "max_tokens": 20,
    "min_tokens": 0,
    "random_seed": None,
    "repetition_penalty": 1.5,
    "stopping_criteria": "length (256 tokens)",
    "stopping_sequence": " "
}

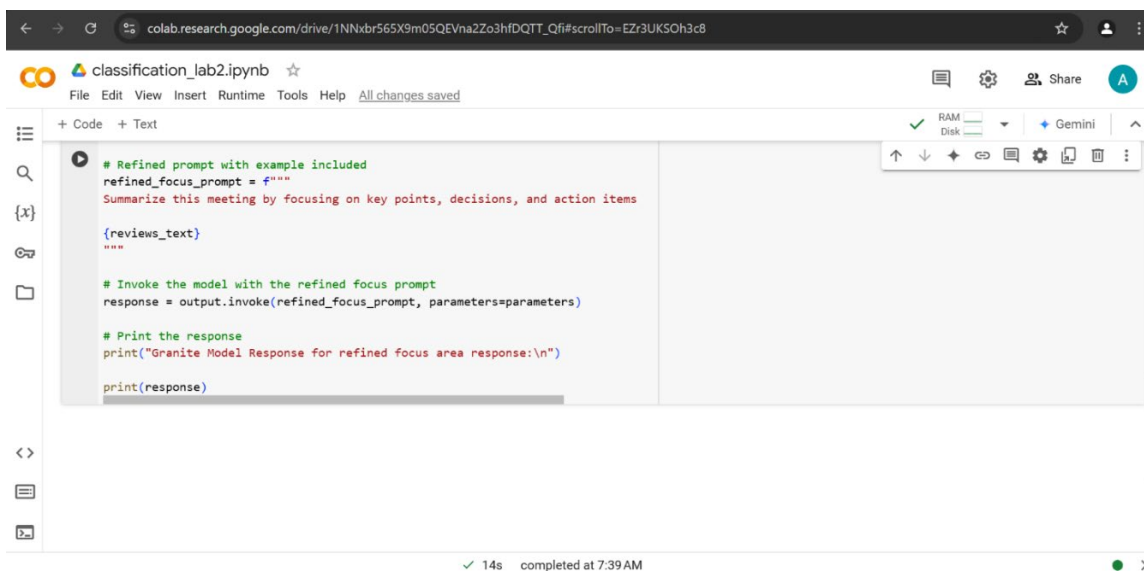
# Define the prompt with refined focus area prompt
refined_focusarea_prompt = f"""
Summarize this meeting by focusing on key points, decisions, and action items

{reviews_text}
"""

response = output.invoke(refined_focusarea_prompt, parameters=parameters)
print("Granite Model Response for refined focus area response:\n")
print(response)
```

✓ 9s completed at 3:03 PM

2. Selecciona el botón **Play** para ejecutar la solicitud.



The screenshot shows a Jupyter Notebook titled 'classification_lab2.ipynb'. The code defines a refined focus prompt and invokes the model with it. The refined focus prompt is defined as a string that instructs the model to summarize a meeting by focusing on key points, decisions, and action items. The code then invokes the model with the refined focus prompt and prints the response.

```
# Refined prompt with example included
refined_focus_prompt = f"""
Summarize this meeting by focusing on key points, decisions, and action items

{reviews_text}
"""

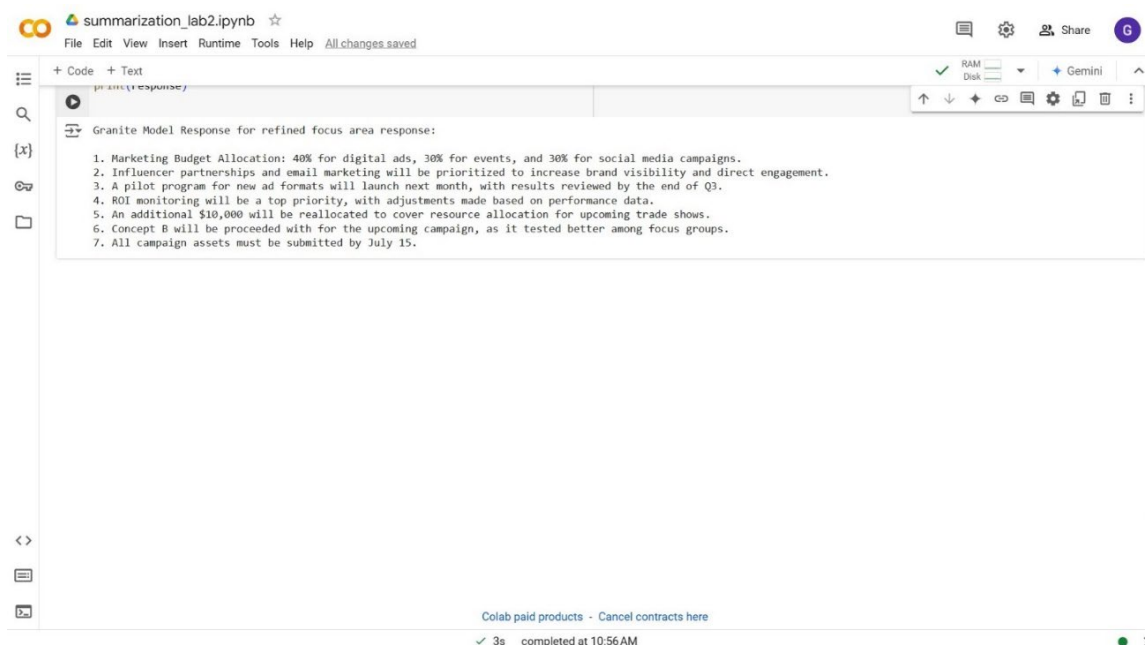
# Invoke the model with the refined focus prompt
response = output.invoke(refined_focus_prompt, parameters=parameters)

# Print the response
print("Granite Model Response for refined focus area response:\n")
print(response)
```

✓ 14s completed at 7:39 AM

3. Lee el resultado que genera el modelo. Ten en cuenta que al ajustar varios parámetros la respuesta fue más concisa, evitando palabras duplicadas. Al establecer `top_k = 10`, el modelo selecciona entre las 10 palabras más probables, controlando el vocabulario y la estructura. Establecer `top_p = 0.9` garantizó que el modelo eligiera palabras que fueran significativas y apropiadas para el contexto. Mantener `repetition_penalty = 1.5` evitó la redacción redundante, mientras que establecer `stop_sequence = " "` detuvo la respuesta en las pausas naturales.

El enfoque equilibrado hacia el ajuste de múltiples parámetros del modelo garantiza que el resultado final sea preciso y adaptado a las necesidades de las partes interesadas.



The screenshot shows a Jupyter Notebook titled 'summarization_lab2.ipynb'. The code cell contains the following Python code:

```
print(response)
```

The output of the code cell is displayed in a text box with the following content:

```
Granite Model Response for refined focus area response:

1. Marketing Budget Allocation: 40% for digital ads, 30% for events, and 30% for social media campaigns.
2. Influencer partnerships and email marketing will be prioritized to increase brand visibility and direct engagement.
3. A pilot program for new ad formats will launch next month, with results reviewed by the end of Q3.
4. ROI monitoring will be a top priority, with adjustments made based on performance data.
5. An additional $10,000 will be reallocated to cover resource allocation for upcoming trade shows.
6. Concept B will be proceeded with for the upcoming campaign, as it tested better among focus groups.
7. All campaign assets must be submitted by July 15.
```

At the bottom of the notebook interface, there is a status bar indicating 'Colab paid products - Cancel contracts here' and a completion message: '✓ 3s completed at 10:56 AM'.

Conclusión

En este laboratorio, utilizaste el modelo **granite-3.0-8b-instruct** y ajustaste los parámetros del modelo para formular la solicitud en dos ciclos a fin de mejorar la calidad del resultado para las tareas de clasificación y resumen. Estos ajustes garantizaron que el resultado fuera consistente, preciso y alineado con las necesidades de las partes interesadas, mejorando la toma de decisiones y la productividad.

© Copyright IBM Corporation 2025.

La información contenida en estos materiales se proporciona únicamente con fines informativos y se proporciona TAL CUAL, sin garantía de ningún tipo, ni expresa ni implícita. IBM no será responsable de ningún daño que surja del uso de estos materiales o que de otro modo esté relacionado con ellos. Nada de lo contenido en estos materiales tiene como propósito, ni tendrá el efecto de, constituir garantías o hacer declaraciones de parte IBM o sus proveedores o licenciantes, ni alterar los términos y condiciones del acuerdo de licencia aplicable que rige el uso del software de IBM. Las referencias que se hagan en estos materiales a productos, programas o servicios de IBM no implican que estarán disponibles en todos los países en los que IBM opera. Esta información se basa en los planes y la estrategia de productos actuales de IBM, que están sujetos a cambios por parte de IBM sin previo aviso. Las fechas de lanzamiento de productos o las capacidades a las que se hace referencia en estos materiales pueden cambiar en cualquier momento a entera discreción de IBM en función de las oportunidades del mercado u otros factores, y no tienen por objeto ser un compromiso con la disponibilidad futura de productos o características de modo alguno.

IBM, el logotipo de IBM e ibm.com son marcas registradas de International Business Machines Corp., registradas en muchas jurisdicciones de todo el mundo. Otros nombres de productos y servicios pueden ser marcas comerciales de IBM o de otras empresas. Una lista actual de las marcas registradas de IBM está disponible en la web en "Información de copyright y marcas registradas" en www.ibm.com/legal/copytrade.shtml.



Please Recycle
