```
# Install required libraries
!pip install -q transformers accelerate bitsandbytes gradio
huggingface_hub
print("✅ Installation complete!")
```

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 59.1/59.1 MB 8.5 MB/s eta
0:00:00
plete!
```

```
import gradio as gr
import time
import random
from transformers import pipeline

print("✅ All libraries imported!")
```

```
✅ All libraries imported!
```

```
from transformers import pipeline
import random

# Use a local model that works without API
generator = pipeline('text-generation', model='gpt2', device=-1)

class DebateAgent:
    def __init__(self, name, role, expertise, stance):
        self.name = name
        self.role = role
        self.expertise = expertise
        self.stance = stance

    def generate_argument(self, topic, conversation_history="",
opponent_argument=""):
        """Generate an argument based on the agent's perspective"""

        prompt = f"{self.name} argues: {self.stance} regarding
{topic}. "

        try:
            # Generate response
            response = generator(
                prompt,
                max_length=100,
                num_return_sequences=1,
                temperature=0.8,
                do_sample=True,
                pad_token_id=50256
            )[0]['generated_text']

            # Extract only the new generated part
            argument = response[len(prompt):].strip()
```

```
            # Clean up the argument
            sentences = argument.split('.')
            argument = '. '.join(sentences[:3]) + '.'

            return argument if argument else f"I believe
{self.stance.lower()} This perspective considers {self.expertise}."

        except Exception as e:
            # Fallback responses based on agent stance
            fallback_args = {
                "Cautious": f"Regarding {topic}, we must proceed
carefully. The risks of moving too quickly without proper safeguards
could have serious consequences. My expertise in {self.expertise}
shows that a measured approach is essential.",
                "Optimistic": f"I believe {topic} presents tremendous
opportunities. Innovation thrives when we embrace change and allow
progress to move forward. Based on my experience in {self.expertise},
rapid advancement benefits everyone.",
                "Balanced": f"Considering {topic}, we need to weigh
both benefits and risks carefully. My work in {self.expertise} has
shown that the best solutions come from thoughtful analysis of
multiple perspectives."
            }

            for key in fallback_args:
                if key.lower() in self.stance.lower():
                    return fallback_args[key]

            return f"From my perspective as a {self.role}, {topic}
requires careful consideration of multiple factors including
{self.expertise}."

# Create three debate agents with different perspectives
def create_agents(topic=""):
    agent1 = DebateAgent(
        name="Dr. Sarah Chen",
        role="AI Ethics Researcher",
        expertise="artificial intelligence safety and ethics",
        stance="Cautious - believes we need strong safeguards and
careful consideration"
    )

    agent2 = DebateAgent(
        name="Marcus Rivera",
        role="Tech Entrepreneur",
        expertise="innovation and business growth",
        stance="Optimistic - believes in rapid innovation and market-
driven solutions"
    )
```

```
    agent3 = DebateAgent(
        name="Prof. Anika Sharma",
        role="Policy Expert",
        expertise="balancing innovation with public interest",
        stance="Balanced - seeks evidence-based middle ground
approaches"
    )

    return [agent1, agent2, agent3]

print("✅ Agent system created successfully!")
print("🚀 Using local model - no API needed!")
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/
_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your
settings tab (https://huggingface.co/settings/tokens), set it as
secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to
access public models or datasets.
  warnings.warn(
```


```
{"model_id":"4012f3886af3409991b0ce51c90c5d7b","version_major":2,"vers
ion_minor":0}
```


```
Warning: You are sending unauthenticated requests to the HF Hub.
Please set a HF_TOKEN to enable higher rate limits and faster
downloads.
WARNING:huggingface_hub.utils._http:Warning: You are sending
unauthenticated requests to the HF Hub. Please set a HF_TOKEN to
enable higher rate limits and faster downloads.
```


```
{"model_id":"f69e5b3d766f4a65bc89bbcb144aca7d","version_major":2,"vers
ion_minor":0}

{"model_id":"0606cf9be9134ea8a8439bb10eab0c69","version_major":2,"vers
ion_minor":0}
```


```
GPT2LMHeadModel LOAD REPORT from: gpt2
Key                   | Status     |  |
----------------------+------------+--+-
h.{0...11}.attn.bias | UNEXPECTED |  |

Notes:
- UNEXPECTED    :can be ignored when loading from different
task/architecture; not ok if you expect identical arch.
```

{"model_id":"2b964d606f1341d99ea9d67071f3ca95","version_major":2,"version_minor":0}

{"model_id":"10add1083cd342929f172640c0505ad7","version_major":2,"version_minor":0}

{"model_id":"a646dad3261643739a57c4d51236c55b","version_major":2,"version_minor":0}

{"model_id":"2a5044fc4afb44319f0e77a8a618661f","version_major":2,"version_minor":0}

{"model_id":"afe342cda79946acbd31c6749efd6983","version_major":2,"version_minor":0}

```
 Agent system created successfully!
 Using local model - no API needed!
```

```python
def run_debate(topic, num_rounds=3):
    """
    Run a multi-agent debate on the given topic
    """
    agents = create_agents()

    debate_output = f" **MULTI-AGENT DEBATE SYSTEM**\n\n"
    debate_output += f" **Topic:** {topic}\n"
    debate_output += f" **Participants:** {agents[0].name}, {agents[1].name}, {agents[2].name}\n"
    debate_output += f"{'='*80}\n\n"

    conversation_history = ""

    for round_num in range(1, num_rounds + 1):
        debate_output += f"### □ Round {round_num}\n\n"

        # Each agent speaks in turn
        for i, agent in enumerate(agents):
            # Get previous agent's argument if available
            previous_arg = ""
            if i > 0:
                previous_arg = agents[i-1].name + "'s argument"

            debate_output += f"**{agent.name}** ({agent.role}):\n"

            # Generate argument
            argument = agent.generate_argument(
                topic=topic,
                conversation_history=conversation_history[:500],  # Keep context manageable
                opponent_argument=previous_arg
            )
```

```
            debate_output += f"{argument}\n\n"

            # Update conversation history
            conversation_history += f"{agent.name}: {argument}\n"

            # Small delay to avoid rate limits
            time.sleep(1)

        debate_output += f"{'-'*80}\n\n"

    # Generate conclusion
    debate_output += "### 🏆 Debate Summary\n\n"
    debate_output += "The debate explored multiple perspectives:\n"
    debate_output += f"- **{agents[0].name}** emphasized cautious,
safety-focused approach\n"
    debate_output += f"- **{agents[1].name}** advocated for innovation
and rapid progress\n"
    debate_output += f"- **{agents[2].name}** sought balanced middle
ground\n\n"
    debate_output += "All perspectives offer valuable insights for
informed decision-making."

    return debate_output

print("✅ Debate orchestrator ready!")
```

```
✅ Debate orchestrator ready!
```

```
def run_debate_enhanced(topic, num_rounds=3):
    """
    Enhanced debate with scoring and analysis
    """
    agents = create_agents()

    debate_output = f"""
# 🎭 MULTI-AGENT DEBATE SYSTEM

## 📋 Topic: {topic}

### 👥 Debate Participants:
- **{agents[0].name}** - {agents[0].role} | Stance: {agents[0].stance}
- **{agents[1].name}** - {agents[1].role} | Stance: {agents[1].stance}
- **{agents[2].name}** - {agents[2].role} | Stance: {agents[2].stance}

---

"""

    conversation_history = ""
    agent_scores = {agent.name: 0 for agent in agents}
```

```
    for round_num in range(1, num_rounds + 1):
        debate_output += f"\n## □ Round {round_num}\n\n"

        round_arguments = []

        # Each agent speaks in turn
        for i, agent in enumerate(agents):
            previous_arg = round_arguments[-1] if round_arguments else
""

            debate_output += f"### □ {agent.name}\n"
            debate_output += f"*{agent.role}*\n\n"

            # Generate argument
            argument = agent.generate_argument(
                topic=topic,
                conversation_history=conversation_history[:300],
                opponent_argument=previous_arg
            )

            debate_output += f"{argument}\n\n"

            # Simple scoring based on argument length and keywords
            score = len(argument.split())
            quality_keywords = ['evidence', 'research', 'data',
'study', 'however', 'therefore', 'because', 'consider']
            score += sum(2 for word in quality_keywords if word in
argument.lower())
            agent_scores[agent.name] += score

            round_arguments.append(argument)
            conversation_history += f"{agent.name}: {argument}\n"

        debate_output += f"\n---\n"

    # Final Analysis
    debate_output += f"\n## □ Debate Analysis\n\n"

    # Sort agents by score
    ranked_agents = sorted(agent_scores.items(), key=lambda x: x[1],
reverse=True)

    debate_output += "### □ Argument Strength Scores:\n\n"
    medals = ["□", "□", "□"]
    for idx, (agent_name, score) in enumerate(ranked_agents):
        medal = medals[idx] if idx < 3 else "□"
        debate_output += f"{medal} **{agent_name}**: {score} points\n"

    debate_output += f"\n### □ Key Takeaways:\n\n"
```

```
    debate_output += f"- **Cautious Perspective**: Emphasized safety
and careful consideration\n"
    debate_output += f"- **Optimistic Perspective**: Highlighted
innovation and opportunities\n"
    debate_output += f"- **Balanced Perspective**: Sought middle
ground with evidence-based approach\n\n"

    debate_output += f"### □ Conclusion:\n\n"
    debate_output += f"This multi-agent debate demonstrated diverse
viewpoints on **{topic}**. "
    debate_output += f"Each agent contributed unique insights based on
their expertise, "
    debate_output += f"showcasing how AI systems can simulate expert
panel discussions for complex decision-making.\n"

    return debate_output

print("□ Enhanced debate system ready!")
```

```
□ Enhanced debate system ready!
```

```
# Create a beautiful Gradio interface
def debate_interface(topic, rounds):
    """Gradio interface function"""
    if not topic.strip():
        return "⚠ Please enter a debate topic!"

    #output = run_debate(topic, int(rounds))
    output = run_debate_enhanced(topic, int(rounds))
    return output

# Build the interface
with gr.Blocks(theme=gr.themes.Soft()) as demo:
    gr.Markdown("""
    # 🤖 Multi-Agent AI Debate System
    ### Watch AI agents debate complex topics from different
perspectives

    This system simulates expert panel discussions where multiple AI
agents
    with different expertise and viewpoints engage in structured
debates.
    """)

    with gr.Row():
        with gr.Column():
            topic_input = gr.Textbox(
                label="Debate Topic",
                placeholder="e.g., Should AI development be
regulated?",
```

```
                lines=2
            )
            rounds_input = gr.Slider(
                minimum=1,
                maximum=5,
                value=3,
                step=1,
                label="Number of Rounds"
            )
            debate_btn = gr.Button("🚀 Start Debate",
variant="primary", size="lg")

    output_box = gr.Markdown(label="Debate Output")

    # Examples
    gr.Examples(
        examples=[
            ["Should AI development be regulated by governments?", 3],
            ["Is remote work better than office work for
productivity?", 3],
            ["Should social media companies be responsible for content
moderation?", 2],
        ],
        inputs=[topic_input, rounds_input]
    )

    debate_btn.click(
        fn=debate_interface,
        inputs=[topic_input, rounds_input],
        outputs=output_box
    )

# Add ability to save debates
def save_debate(debate_text, topic):
    """Save debate to a text file"""
    filename = f"debate_{topic[:30].replace(' ', '_')}.txt"
    with open(filename, 'w', encoding='utf-8') as f:
        f.write(debate_text)
    return f"✅ Debate saved as {filename}"

print("✅ Export feature added!")
print("🎉 Interface created!")
print("🚀 Launching demo...")

# Launch the interface
demo.launch(share=True, debug=True)
```

```
/tmp/ipython-input-1890147794.py:12: DeprecationWarning: The 'theme'
parameter in the Blocks constructor will be removed in Gradio 6.0. You
```

```
will need to pass 'theme' to Blocks.launch() instead.
  with gr.Blocks(theme=gr.themes.Soft()) as demo:

□ Export feature added!
□ Interface created!
□ Launching demo...
Colab notebook detected. This cell will run indefinitely so that you
can see errors and logs. To turn off, set debug=False in launch().
* Running on public URL: https://136de1af28026293dc.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU
upgrades, run `gradio deploy` from the terminal in the working
directory to deploy to Hugging Face Spaces
(https://huggingface.co/spaces)

<IPython.core.display.HTML object>

Passing `generation_config` together with generation-related
arguments=({'temperature', 'max_length', 'num_return_sequences',
'pad_token_id', 'do_sample'}) is deprecated and will be removed in
future versions. Please pass either a `generation_config` object OR
all generation parameters explicitly, but not both.
Both `max_new_tokens` (=256) and `max_length`(=100) seem to have been
set. `max_new_tokens` will take precedence. Please refer to the
documentation for more information.
(https://huggingface.co/docs/transformers/main/en/main_classes/text_ge
neration)
Both `max_new_tokens` (=256) and `max_length`(=100) seem to have been
set. `max_new_tokens` will take precedence. Please refer to the
documentation for more information.
(https://huggingface.co/docs/transformers/main/en/main_classes/text_ge
neration)
Both `max_new_tokens` (=256) and `max_length`(=100) seem to have been
set. `max_new_tokens` will take precedence. Please refer to the
documentation for more information.
(https://huggingface.co/docs/transformers/main/en/main_classes/text_ge
neration)
Both `max_new_tokens` (=256) and `max_length`(=100) seem to have been
set. `max_new_tokens` will take precedence. Please refer to the
documentation for more information.
(https://huggingface.co/docs/transformers/main/en/main_classes/text_ge
neration)
Both `max_new_tokens` (=256) and `max_length`(=100) seem to have been
set. `max_new_tokens` will take precedence. Please refer to the
documentation for more information.
(https://huggingface.co/docs/transformers/main/en/main_classes/text_ge
neration)
Both `max_new_tokens` (=256) and `max_length`(=100) seem to have been
set. `max_new_tokens` will take precedence. Please refer to the
documentation for more information.
```

```
(https://huggingface.co/docs/transformers/main/en/main_classes/text_ge
neration)
Both `max_new_tokens` (=256) and `max_length`(=100) seem to have been
set. `max_new_tokens` will take precedence. Please refer to the
documentation for more information.
(https://huggingface.co/docs/transformers/main/en/main_classes/text_ge
neration)
Both `max_new_tokens` (=256) and `max_length`(=100) seem to have been
set. `max_new_tokens` will take precedence. Please refer to the
documentation for more information.
(https://huggingface.co/docs/transformers/main/en/main_classes/text_ge
neration)
Both `max_new_tokens` (=256) and `max_length`(=100) seem to have been
set. `max_new_tokens` will take precedence. Please refer to the
documentation for more information.
(https://huggingface.co/docs/transformers/main/en/main_classes/text_ge
neration)
Both `max_new_tokens` (=256) and `max_length`(=100) seem to have been
set. `max_new_tokens` will take precedence. Please refer to the
documentation for more information.
(https://huggingface.co/docs/transformers/main/en/main_classes/text_ge
neration)
Both `max_new_tokens` (=256) and `max_length`(=100) seem to have been
set. `max_new_tokens` will take precedence. Please refer to the
documentation for more information.
(https://huggingface.co/docs/transformers/main/en/main_classes/text_ge
neration)
Both `max_new_tokens` (=256) and `max_length`(=100) seem to have been
set. `max_new_tokens` will take precedence. Please refer to the
documentation for more information.
(https://huggingface.co/docs/transformers/main/en/main_classes/text_ge
neration)
Both `max_new_tokens` (=256) and `max_length`(=100) seem to have been
set. `max_new_tokens` will take precedence. Please refer to the
documentation for more information.
(https://huggingface.co/docs/transformers/main/en/main_classes/text_ge
neration)
Both `max_new_tokens` (=256) and `max_length`(=100) seem to have been
set. `max_new_tokens` will take precedence. Please refer to the
documentation for more information.
(https://huggingface.co/docs/transformers/main/en/main_classes/text_ge
neration)
Both `max_new_tokens` (=256) and `max_length`(=100) seem to have been
set. `max_new_tokens` will take precedence. Please refer to the
documentation for more information.
(https://huggingface.co/docs/transformers/main/en/main_classes/text_ge
neration)
Both `max_new_tokens` (=256) and `max_length`(=100) seem to have been
set. `max_new_tokens` will take precedence. Please refer to the
```

```
documentation for more information.
(https://huggingface.co/docs/transformers/main/en/main_classes/text_ge
neration)
Both `max_new_tokens` (=256) and `max_length`(=100) seem to have been
set. `max_new_tokens` will take precedence. Please refer to the
documentation for more information.
(https://huggingface.co/docs/transformers/main/en/main_classes/text_ge
neration)
Both `max_new_tokens` (=256) and `max_length`(=100) seem to have been
set. `max_new_tokens` will take precedence. Please refer to the
documentation for more information.
(https://huggingface.co/docs/transformers/main/en/main_classes/text_ge
neration)
Both `max_new_tokens` (=256) and `max_length`(=100) seem to have been
set. `max_new_tokens` will take precedence. Please refer to the
documentation for more information.
(https://huggingface.co/docs/transformers/main/en/main_classes/text_ge
neration)
Both `max_new_tokens` (=256) and `max_length`(=100) seem to have been
set. `max_new_tokens` will take precedence. Please refer to the
documentation for more information.
(https://huggingface.co/docs/transformers/main/en/main_classes/text_ge
neration)
Both `max_new_tokens` (=256) and `max_length`(=100) seem to have been
set. `max_new_tokens` will take precedence. Please refer to the
documentation for more information.
(https://huggingface.co/docs/transformers/main/en/main_classes/text_ge
neration)
```