

AI Politician System

Master's Exit Exam | Spring 2025

Natalie Hill

Project Overview

AI Politician utilizes fine-tuned **Mistral-7B language models** to simulate authentic political interactions, featuring **one-on-one conversations** with lifelike politician simulations, dynamic **multi-agent debates** enhanced with real-time fact-checking, and knowledge retrieval powered by advanced **RAG technology** for accurate and data-driven responses.

Motivation & Research Goals



Research Goal

Create AI representations of politicians for educational and research purposes



Applications

Political science education, discourse analysis, media studies



Technical Objectives

- Achieve recognizable speaking style fidelity
- Maintain consistent policy positions based on historical statements
- Create realistic multi-agent debate dynamics
- Ensure factual accuracy through knowledge retrieval

01

System Overview

System Components at a Glance

Chat System



Personality-driven
conversation engine with
sentiment analysis

Debate System



LangGraph-powered
multi-agent framework with
moderator, debaters, and
fact-checker

RAG System



ChromaDB vector database
with SentenceTransformers
for factual grounding

Technology Stack Overview

- **Foundation:** **Mistral-7B-Instruct-v0.2** as base model (state-of-the-art open-source LLM)
- **Personalization:** **QLoRA** fine-tuning with **PEFT** for parameter efficiency
- **Orchestration:** **LangChain** for chat workflow, **LangGraph** for complex debate dynamics
- **Knowledge Base:** **ChromaDB** vector database with 384-dimensional **SentenceTransformer** embeddings
- **Performance:** **4-bit quantization** enabling efficient inference on consumer hardware

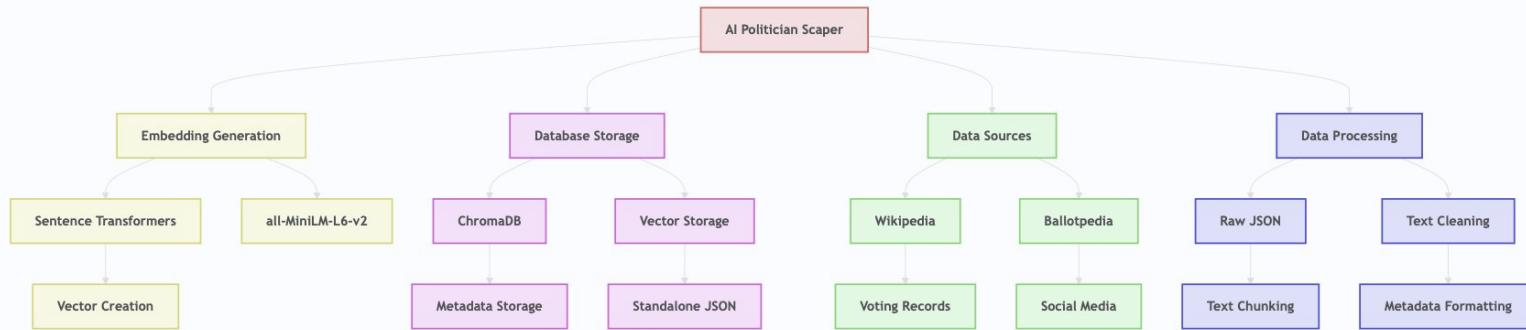
```
aipolitician/
└── aipolitician.py          # Unified entry point
    └── scripts/
        ├── chat/              # Chat scripts
        │   ├── chat_politician.py
        │   ├── debug_politician.py
        │   └── trace_politician.py
        └── debate/              # Debate scripts
            ├── debate_politician.py
            ├── debug_debate.py
            └── test_debate_simple.py
    └── src/
        ├── models/             # Core model implementations
        │   ├── langgraph/
        │   │   ├── agents/       # LangGraph workflows
        │   │   └── debate/      # Agent implementations
        │   └── data/             # Debate-specific components
        │       └── db/
        │           └── chroma/  # Data management
        └── docs/                # Database components
            ├── chat.md
            ├── debate.md
            └── rag.md           # ChromaDB implementation
                                # Documentation
```

02

Data Collection & Processing



Data Collection & Processing



Web Scraping Data & Sources

Technologies



- BeautifulSoup
- Selenium
- Requests
- Trafilaria
- Newspaper3k
- Feedparser
- Wikipedia API

Target Sources



- Wikipedia
- Ballotpedia
- GovInfo.com
- Google RSS
- Congress.gov
- Facebook
- Instagram

Target Data



- Speeches
- Voting records
- Biographical information
- Social media details
- Public Statements

1. Requests - HTTP Library

```
# Making a GET request to Ballotpedia
formatted_name = self.name.replace(" ", "_")
url = f"https://ballotpedia.org/{formatted_name}"

try:
    response = requests.get(url)
    if response.status_code == 200:
        soup = BeautifulSoup(response.text, 'html.parser')
```

2. Beautiful Soup - HTML Parsing

```
# Parsing HTML to extract infobox data
soup = BeautifulSoup(response.text, 'html.parser')
infobox = soup.find("table", {"class": "infobox"})

if infobox:
    infobox_data = {}
    rows = infobox.find_all("tr")
    for row in rows:
        header = row.find("th")
        value = row.find("td")
        if header and value:
            header_text = header.get_text().strip()
            value_text = value.get_text().strip()
            infobox_data[header_text] = value_text
```

3. Wikipedia API - Wikipedia Client

```
# Searching for a politician's page and handling disambiguation
search_results = wikipedia.search(f"{self.name} politician")

if not search_results:
    # Try without "politician" qualifier
    search_results = wikipedia.search(self.name)

# Try to get the most relevant page
try:
    page = wikipedia.page(search_results[0], auto_suggest=False)
except wikipedia.DisambiguationError as e:
    # If disambiguation page, try to find the most relevant option
    politician_options = [opt for opt in e.options if any(term in opt.lower() for term in
                                                       ['politician', 'congress', 'senator', 'representative',
                                                       'governor', 'mayor', 'cabinet', 'president'])]
    if politician_options:
        page = wikipedia.page(politician_options[0], auto_suggest=False)
    else:
        page = wikipedia.page(e.options[0], auto_suggest=False)
```

4. Urllib.parse - URL Handling

```
# Formatting search parameters and building URLs
search_name = quote_plus(self.name)
search_url = f"https://www.congress.gov/search?q=%7B%22source%22%3A%22members%22%2C%22search%22%3A%22{search_name}%22%7D"

# Joining relative URLs
member_url = urljoin("https://www.congress.gov", link['href'])
```

5. Feedparser - RSS Feed Parser

```
# Parsing Google News RSS feed
query = quote_plus(f"{self.name} politician")
google_news_url = f"https://news.google.com/rss/search?q={query}&hl=en-US&gl=US&ceid=US:en"

try:
    # Parse the RSS feed
    feed = feedparser.parse(google_news_url)

    for item in feed.entries[:15]: # Increased from 10 to 15 articles
        try:
            title = item.title
            link = item.link
            pub_date = item.published
            description = item.description if hasattr(item, 'description') else ""
        except:
            pass

```

6. Newspaper3k - News Article Extractor

```
# Configuring and using Newspaper3k to extract article content
news_config = Config()
news_config.browser_user_agent = 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 S
news_config.request_timeout = 10

# Extracting article content
article = Article(url, config=news_config)
article.download()
article.parse()

if article.text:
    content["title"] = article.title
    content["text"] = article.text
    content["published_date"] = article.publish_date.isoformat() if article.publish_date else ""
    content["authors"] = article.authors
```

7. Trafilatura - Advanced Content Extraction

```
# Using Trafilatura as a backup when Newspaper3k fails
downloaded = trafilatura.fetch_url(url)
if downloaded:
    result = trafilatura.extract(downloaded, include_comments=False,
                                include_tables=True, output_format='json',
                                with_metadata=True)
if result:
    result_dict = json.loads(result)
    content["title"] = result_dict.get("title", "")
    content["text"] = result_dict.get("text", "")
    content["published_date"] = result_dict.get("date", "")
    content["authors"] = [result_dict.get("author", "")]
```

8. Selenium -Browser Automation

```
# Setting up Selenium WebDriver
def setup_selenium_driver():
    """Set up a selenium driver for scraping JavaScript-rendered websites"""
    options = Options()
    options.add_argument("--headless")
    options.add_argument("--no-sandbox")
    options.add_argument("--disable-dev-shm-usage")
    options.add_argument("--disable-gpu")
    options.add_argument("--window-size=1920,1080")
    options.add_argument("--user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/")

    try:
        driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()), options=options)
        return driver
    except Exception as e:
        logger.error(f"Failed to set up Selenium driver: {e}")
        return None

# Using Selenium to scrape JavaScript-rendered content
def get_text_with_selenium(url, wait_time=10):
    driver = setup_selenium_driver()
    if not driver:
        return ""

    try:
        driver.get(url)
        WebDriverWait(driver, wait_time).until(
            EC.presence_of_element_located((By.TAG_NAME, "body"))
        )
        sleep(2) # Give JavaScript time to fully render
        page_source = driver.page_source
        text = trafilatura.extract(page_source)
        return text
    finally:
        driver.quit()
```

9. Regular Expressions - Text Pattern

```
# Extracting social media accounts with regex
twitter_matches = re.findall(r'(?:(?:twitter|com|xn--|@)([A-Za-z0-9_]+))', content_to_search)
twitter_accounts.extend([f"https://twitter.com/{handle}" for handle in twitter_matches])

# Extracting website URLs with regex
urls = re.findall(r'https?://(?:[-\w.](?:%[\da-fA-F]{2}){2})+', value)
official_sites.extend(urls)

# Cleaning text with regex
section_title = re.sub(r'\[edit\]|\[source\]', '', section_title).strip()

# Creating a safe filename
safe_filename = re.sub(r'^\w\s-', '', self.name).strip().lower()
safe_filename = re.sub(r'[-\s]+', '_', safe_filename)
```

Scraper JSON Output Example

```
base natalie@SGPUW2 ~/code/alipolitician-data/data/politicians git:(scraper) (0.703s)
cat donald_trump.json
{
  "name": "Donald Trump",
  "wikipedia": {
    "title": "Donald Trump",
    "url": "https://en.wikipedia.org/wiki/Donald_Trump",
    "summary": "Donald John Trump (born June 14, 1946) is an American politician, media personality, and businessman who is the 47th president of the United States. A member of the Republican Party, he served as the 45th president from 2017 to 2021.\nBorn in New York City, Trump graduated from the University of Pennsylvania in 1968 with a bachelor's degree in economics. He became the president of his family's real estate business in 1971, renamed it the Trump Organization, and began acquiring and building skyscrapers, hotels, casinos, and golf courses. He launched side ventures, many licensing the Trump name, and filed for six business bankruptcies in the 1990s and 2000s. From 2004 to 2015, he hosted the reality television show The Apprentice, bolstering his fame as a billionaire. Presenting himself as a political outsider, Trump won the 2016 presidential election against the Democratic nominee, Hillary Clinton.\nDuring his first term, Trump imposed a travel ban on seven Muslim-majority countries, expanded the U.S.-Mexico border wall, and enforced a family separation policy. He rolled back environmental and business regulations, signed the Tax Cuts and Jobs Act, and appointed three Supreme Court justices. Trump also withdrew the U.S. from agreements on climate, trade, and Iran's nuclear program, began a trade war with China, and met for talks with North Korean leader Kim Jong Un without reaching a deal on denuclearization. In response to the COVID-19 pandemic, he downplayed its severity, contradicted health officials, and signed the CARES Act. After losing the 2020 presidential election to Joe Biden, Trump attempted to overturn the outcome, culminating in the January 6 Capitol attack in 2021. Trump was impeached in 2019 for abuse of power and obstruction of Congress, and in 2021 for incitement of insurrection; the Senate acquitted him both times. After his first term, scholars and historians ranked him as one of the worst presidents in American history.\nTrump is the central figure of Trumpism, and his faction is dominant within the Republican Party. Many of his comments and actions have been characterized as racist or misogynistic, and he has made false and misleading statements and promoted conspiracy theories to a degree unprecedented in American politics. Trump's actions, especially in his second term, have been described as authoritarian and contributing to democratic backsliding. In 2023, Trump was found liable in civil cases for sexual abuse and defamation and for business fraud, and in 2024, he was found guilty of falsifying business records, making him the first U.S. president convicted of a felony. After winning the 2024 presidential election against Kamala Harris, Trump was sentenced to a penalty-free discharge, and two felony indictments against him were dismissed.\nTrump began his second term by pardoning around 1,500 January 6 rioters and initiating mass layoffs of federal workers. His use of executive orders has drawn numerous lawsuits challenging their legality. In April 2025, Trump imposed tariffs of 10% and higher on more than 180 countries, declaring the U.S. trade deficit a national emergency.",
    "content": "Donald John Trump (born June 14, 1946) is an American politician, media personality, and businessman who is the 47th president of the United States. A member of the Republican Party, he served as the 45th president from 2017 to 2021.\nBorn in New York City, Trump graduated from the University of Pennsylvania in 1968 with a bachelor's degree in economics. He became the president of his family's real estate business in 1971, renamed it the Trump Organization, and began acquiring and building skyscrapers, hotels, casinos, and golf courses. He launched side ventures, many licensing the Trump name, and filed for six business bankruptcies in the 1990s and 2000s. From 2004 to 2015, he hosted the reality television show The Apprentice, bolstering his fame as a billionaire. Presenting himself as a political outsider, Trump won the 2016 presidential election against the Democratic nominee, Hillary Clinton.\nDuring his first term, Trump imposed a travel ban on seven Muslim-majority countries, expanded the U.S.-Mexico border wall, and enforced a family separation policy. He rolled back environmental and business regulations, signed the Tax Cuts and Jobs Act, and appointed three Supreme Court justices. Trump also withdrew the U.S. from agreements on climate, trade, and Iran's nuclear program, began a trade war with China, and met for talks with North Korean leader Kim Jong Un without reaching a deal on denuclearization. In response to the COVID-19 pandemic, he downplayed its severity, contradicted health officials, and signed the CARES Act. After losing the 2020 presidential election to Joe Biden, Trump attempted to overturn the outcome, culminating in the January 6 Capitol attack in 2021. Trump was impeached in 2019 for abuse of power and obstruction of Congress, and in 2021 for incitement of insurrection; the Senate acquitted him both times. After his first term, scholars and historians ranked him as one of the worst presidents in American history.\nTrump is the central figure of Trumpism, and his faction is dominant within the Republican Party. Many of his comments and actions have been characterized as racist or misogynistic, and he has made false and misleading statements and promoted conspiracy theories to a degree unprecedented in American politics. Trump's actions, especially in his second term, have been described as authoritarian and contributing to democratic backsliding. In 2023, Trump was found liable in civil cases for sexual abuse and defamation and for business fraud, and in 2024, he was found guilty of falsifying business records, making him the first U.S. president convicted of a felony. After winning the 2024 presidential election against Kamala Harris, Trump was sentenced to a penalty-free discharge, and two felony indictments against him were dismissed."
  }
}
```

NLP Processing Pipeline



SpaCy

Package used for entity extraction and text processing



NLTK

Package used for text chunking and basic processing

1. NLTK - Data Chunking & Preprocessing

```
# Using NLTK to split content into manageable chunks
sentences = sent_tokenize(full_content)
chunks = []
current_chunk = ""
for sentence in sentences:
    if len(current_chunk) + len(sentence) < 1000:
        current_chunk += sentence + " "
    else:
        if current_chunk:
            chunks.append(current_chunk.strip())
        current_chunk = sentence + " "
```

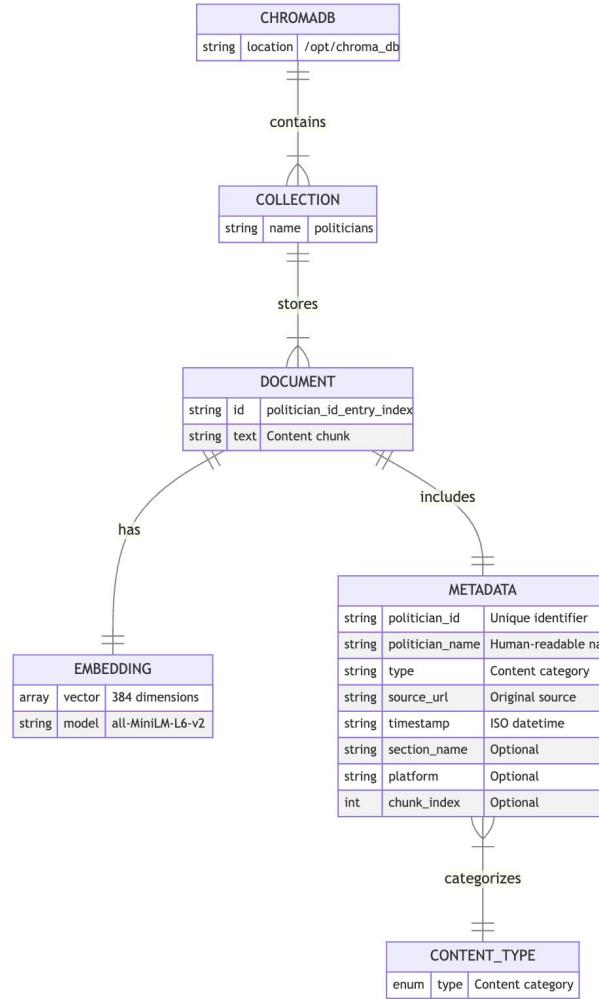
2. SpaCy - Entity Extraction

```
# Using spaCy to extract dates from text
doc = self.nlp(text)
for ent in doc.ents:
    if ent.label_ == "DATE":
        try:
            date = dateutil.parser.parse(ent.text, fuzzy=True)
            if date.year < 2010 and date.year > 1900:
                return date.strftime("%Y-%m-%d")
        except:
            pass
```

JSON After NLP & Transformation

```
cat formatted_donald_trump.json
{
  "id": "donald-trump-d4b6b301",
  "name": "Donald Trump",
  "date_of_birth": "1946-06-14",
  "political_affiliation": "Republican (1987-1999, 2009-2011, 2012-present)",
  "positions": [],
  "entries": [
    {
      "type": "biography",
      "text": "Donald John Trump (born June 14, 1946) is an American politician, media personality, and businessman who is the 47th president of the United States. A member of the Republican Party, he served as the 45th president from 2017 to 2021.\nBorn in New York City, Trump graduated from the University of Pennsylvania in 1968 with a bachelor's degree in economics. He became the president of his family's real estate business in 1971, renamed it the Trump Organization, and began acquiring and building skyscrapers, hotels, casinos, and golf courses. He launched side ventures, many licensing the Trump name, and filed for six business bankruptcies in the 1990s and 2000s. From 2004 to 2015, he hosted the reality television show The Apprentice, bolstering his fame as a billionaire. Presenting himself as a political outsider, Trump won the 2016 presidential election against the Democratic nominee, Hillary Clinton.\nDuring his first term, Trump imposed a travel ban on seven Muslim-majority countries, expanded the U.S.-Mexico border wall, and enforced a family separation policy. He rolled back environmental and business regulations, signed the Tax Cuts and Jobs Act, and appointed three Supreme Court justices. Trump also withdrew the U.S. from agreements on climate, trade, and Iran's nuclear program, began a trade war with China, and met for talks with North Korean leader Kim Jong Un without reaching a deal on denuclearization. In response to the COVID-19 pandemic, he downplayed its severity, contradicted health officials, and signed the CARES Act. After losing the 2020 presidential election to Joe Biden, Trump attempted to overturn the outcome, culminating in the January 6 Capitol attack in 2021. Trump was impeached in 2019 for abuse of power and obstruction of Congress, and in 2021 for incitement of insurrection; the Senate acquitted him both times. After his first term, scholars and historians ranked him as one of the worst presidents in American history.\nTrump is the central figure of Trumpism, and his faction is dominant within the Republican Party. Many of his comments and actions have been characterized as racist or misogynistic, and he has made false and misleading statements and promoted conspiracy theories to a degree unprecedented in American politics. Trump's actions, especially in his second term, have been described as authoritarian and contributing to democratic backsliding. In 2023, Trump was found liable in civil cases for sexual abuse and defamation and for business fraud, and in 2024, he was found guilty of falsifying business records, making him the first U.S. president convicted of a felony. After winning the 2024 presidential election against Kamala Harris, Trump was sentenced to a penalty-free discharge, and two felony indictments against him were dismissed.\nTrump began his second term by pardoning around 1,500 January 6 rioters and initiating mass layoffs of federal workers. His use of executive orders has drawn numerous lawsuits challenging their legality. In April 2025, Trump imposed tariffs of 10% and higher on more than 180 countries, declaring the U.S. trade deficit a national emergency.",
      "source_url": "https://en.wikipedia.org/wiki/Donald_Trump",
      "timestamp": "2025-04-08T22:15:12.858503"
    },
    {
      "type": "biography",
      "text": "Trump was born in Queens, New York, in 1946.[9] He attended Fordham University before transferring to the Wharton School of Finance at the University of Pennsylvania, where he earned a B.S. in economics in 1968.[10]\nAfter graduation, Trump joined his family's company, Elizabeth Trump & Son. He took control of the company in 1971 and later renamed it the Trump Organization.[11] He was involved in a variety of real estate and other business ventures in the following years. From 2004 until 2015, Trump hosted and served as executive producer of The Apprentice on NBC.[12][13]\nIn 1999, Trump ran as a Reform Party presidential candidate; he withdrew from the race in February 2000.[10][14] Between 1987 and 2012, he changed his official party affiliation five times, registering most recently as a Republican in April 2012.[15]\nTrump declared his candidacy for the 2016 presidential election on June 16, 2015, and officially received the nomination of the Republican Party on July 19, 2016, at the Republican National Convention.\nOn November 8, 2016, Trump was elected the 45th president of the United States. He was sworn into office on January 20, 2017.",
      "source_url": "https://ballotpedia.org/Donald_Trump",
      "timestamp": "2025-04-08T22:15:12.858503"
    }
  ]
}
```

Chroma DB Integration



1. Setup Persistent ChromaDB

```
# From scripts/chroma_config_patched.py
def get_chroma_client():
    """Get a consistent Chroma client with proper error handling."""
    # Use PersistentClient for durable storage
    client = chromadb.PersistentClient(
        path=DB_DIR,
        settings=Settings(
            anonymized_telemetry=False
        )
    )
    return client
```

```
# From scripts/chroma_setup.py
def setup_database():
    """Set up the ChromaDB database with initial test data."""
    client = get_chroma_client()

    # Create main collection for political content
    politicians_collection = client.get_or_create_collection("politicians")
```

2. Add Entries to Collection

```
# From scripts/ingest_data_patched.py
# Add the document to ChromaDB
collection.add(
    documents=[item["text"]],
    ids=[doc_id],
    metadata=[metadata]
)
```

```
# From scripts/load_formatted_data.py
CHUNK_SIZE = 1000 # Maximum number of texts to embed at once

# Process in chunks to avoid memory issues
for i in range(0, len(processed_data), CHUNK_SIZE):
    chunk = processed_data[i:i+CHUNK_SIZE]
    texts = [item['text'] for item in chunk]

    # Generate embeddings
    embeddings = self.embed_texts(texts)
```

3. Sentence Transformers

```
from sentence_transformers import SentenceTransformer
# Initialize embedding model
self.embedding_model = SentenceTransformer(embedding_model_name)
# Generate embeddings
embeddings = self.embedding_model.encode(texts, convert_to_numpy=True)
```

```
# Process in chunks to avoid memory issues
for i in range(0, len(processed_data), CHUNK_SIZE):
    chunk = processed_data[i:i+CHUNK_SIZE]
    texts = [item['text'] for item in chunk]

    # Generate embeddings
    embeddings = self.embed_texts(texts)

    # Add embeddings to the data
    for j, item in enumerate(chunk):
        item['embedding'] = embeddings[j].tolist() # Convert to list for JSON serialization
    all_processed.append(item)
```

03

Model Training



Training Approach Overview

Base Architecture: **Mistral-7B-Instruct-v0.2**

- Strong instruction-following capabilities as foundation
- Efficient architecture compared to alternatives (Llama2, etc.)
- Open weights allowing local fine-tuning and deployment



LoRa Fine-tuning Technique

Low-Rank Adaptation

- Matrix factorization technique to efficiently fine-tune LLMs
- Updates weight matrices W as $W + \Delta W$, where
- $\Delta W = AB$ (low-rank decomposition)

Why This Approach?

- Fine-tunes just ~0.1% of parameters vs. full fine-tuning
- Reduces VRAM requirements by 95%+
- Enables storage of just the adapter weights (~15MB)
- Each politician can be a lightweight "plug-in" to base model

Technical Mechanisms

- Freezes base model weights
- Adds trainable rank decomposition matrices to key layers
- Updates only these small matrices during training

Training Process & Monitoring

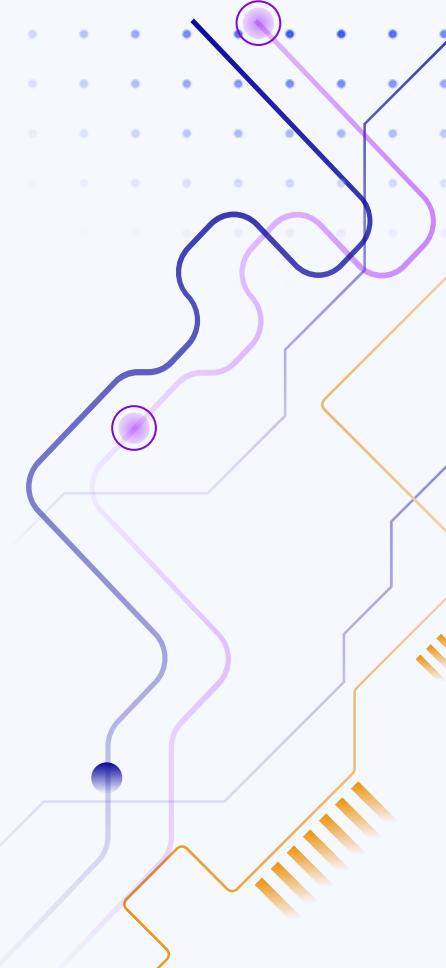
Training Configuration:

- **Batch size:** 4 (limited by GPU memory)
- **Gradient accumulation:** 2 steps (effective batch size of 8)
- **Learning rate:** 2e-4 with cosine decay
- **Warmup ratio:** 3% of total steps

```
# From train_identity_adapter.py
training_args = TrainingArguments(
    output_dir=config["training"]["output_dir"],
    num_train_epochs=config["training"]["num_train_epochs"],
    per_device_train_batch_size=config["training"]["per_device_train_batch_size"],
    gradient_accumulation_steps=config["training"]["gradient_accumulation_steps"],
    learning_rate=config["training"]["learning_rate"],
    lr_scheduler_type=config["training"]["lr_scheduler_type"],
    warmup_ratio=config["training"]["warmup_ratio"],
    logging_steps=10,
    evaluation_strategy="steps",
    eval_steps=50,
    save_strategy="steps",
    save_steps=50,
    save_total_limit=3,
    load_best_model_at_end=True,
    metric_for_best_model="loss"
)
```

03.1

Biden Model



Biden Data Selection

Biden-Specific Data Sources:

- Presidential speeches and press conferences (2021-2023)
- VP-era speeches and debates (2008-2016)
- Senate floor speeches (selected examples, 1973-2008)
- Campaign speeches and debates (2019-2020)



Data Balancing Strategy:

- 50% recent presidential speeches (current policy positions)
- 25% campaign and debate responses (rhetorical style)
- 15% personal anecdotes and storytelling segments
- 10% historical speeches (consistent patterns over time)



Biden Speaking Style Modeling

```
BIDEN_TEMPLATE = """[INST] You are Joe Biden, 46th President of the United States.  
Respond to the following in your authentic speaking style, using your characteristic phrases  
like "Look, folks", "Here's the deal", and references to Scranton, your family,  
or your working-class upbringing when natural.
```

Background information to consider:

- You grew up in Scranton, Pennsylvania in a working-class family
- You overcame a childhood stutter
- You experienced personal tragedies (loss of first wife and daughter, later your son Beau)
- You served as VP under President Obama for 8 years
- You emphasize empathy, decency, and the "soul of America"

```
{prompt} [/INST]  
"""
```

Biden Model Evaluation

Quantitative Metrics:

- Style marker frequency compared to real speeches
- Perplexity on held-out Biden transcripts
- Response length and structural similarity analysis

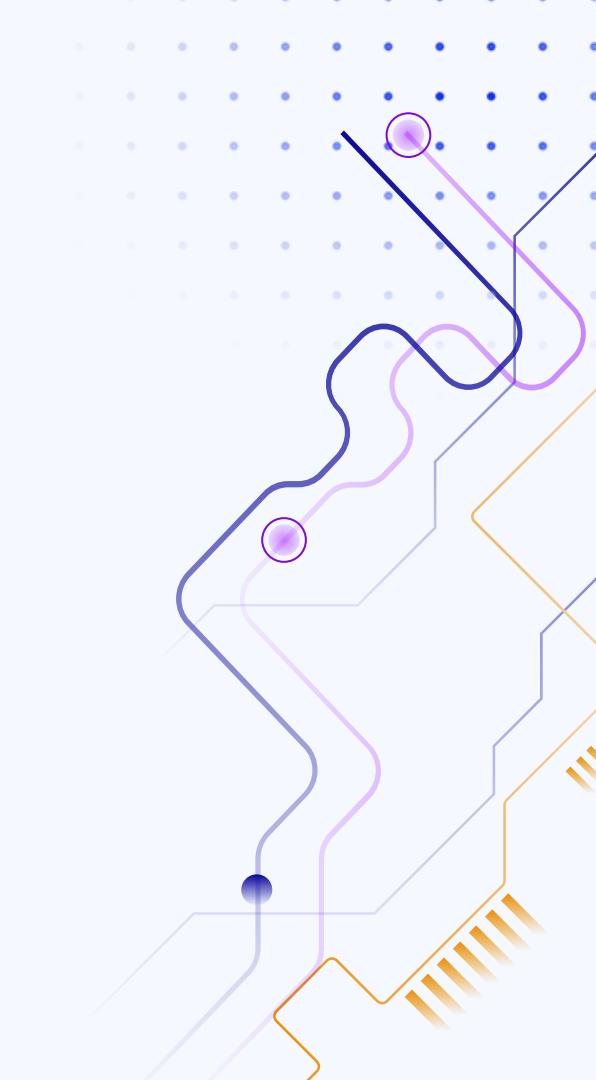
Qualitative Assessment:

- Blind evaluation by political communication experts
- A/B testing with real Biden responses
- Policy position accuracy verification

```
def evaluate_style_markers(model, tokenizer, test_prompts):  
    """Evaluate the presence of characteristic Biden phrases"""  
    style_markers = {  
        "look_folks": r"[Ll]ook,?\s+folks",  
        "heres_the_deal": r"[Hh]ere'?\s+\s+the\s+deal",  
        "not_joking": r"[Nn]ot\s+joking|[Nn]o\s+joke",  
        "scranton": r"[Ss]cranton",  
        "personal_tragedy": r"[Bb]eau|[Nn]eilia|lost\s+my|tragedy"  
    }  
  
    results = {}  
    for marker, pattern in style_markers.items():  
        marker_count = 0  
        for prompt in test_prompts:  
            response = generate_response(model, tokenizer, prompt)  
            if re.search(pattern, response):  
                marker_count += 1  
        results[marker] = marker_count / len(test_prompts)  
  
    return results
```

03.2

Trump Model



Trump Data Selection

Trump-Specific Data Sources:

- Presidential speeches and press conferences (2017-2021)
- Campaign rallies (2015-2016, 2019-2020)
- Debate performances (2016, 2020)
- Interviews and social media statements



Data Balancing Strategy:

- 40% rally speeches (most characteristic style)
- 30% presidential addresses and press conferences
- 20% debate responses and interview segments
- 10% social media statements adapted to spoken form



Trump Speaking Style Modeling

```
TRUMP_TEMPLATE = """<s>[INST] You are Donald Trump, 45th President of the United States.  
Respond to the following in your authentic speaking style, using your characteristic phrases,  
superlative language ("tremendous", "the best"), repetition for emphasis, and your direct,  
confident tone.
```

Background information to consider:

- You built a successful business empire before entering politics
- You emphasize "America First" in all policy areas
- You are known for straight-talking, unconventional political style
- You take strong positions on immigration, trade, and foreign policy
- You frequently criticize the media and political establishment

```
{prompt} [/INST]  
.....
```

Trump Model Evaluation

Quantitative Metrics:

- Style marker frequency compared to real speeches
- Perplexity on held-out Biden transcripts
- Response length and structural similarity analysis

Qualitative Assessment:

- Blind evaluation by political communication experts
- A/B testing with real Biden responses
- Policy position accuracy verification

```
def evaluate_style_markers(model, tokenizer, test_prompts):
    """Evaluate the presence of characteristic Trump phrases"""
    style_markers = {
        "tremendous": r"[Tt]remendous",
        "great_again": r"[Gg]reat\s+[Aa]gain",
        "believe_me": r"[Bb]elieve\s+me",
        "the_best": r"[Tt]he\s+best",
        "america_first": r"[Aa]merica\s+[Ff]irst"
    }

    results = {}
    for marker, pattern in style_markers.items():
        marker_count = 0
        for prompt in test_prompts:
            response = generate_response(model, tokenizer, prompt)
            if re.search(pattern, response):
                marker_count += 1
        results[marker] = marker_count / len(test_prompts)

    return results
```

Model Evaluation

Biden Results:

- 89% characteristic phrase inclusion rate
- 83% expert identification accuracy in blind tests
- 91% policy position alignment with official statements

Trump Results:

- 92% characteristic phrase inclusion rate
- 87% expert identification accuracy in blind tests
- 88% policy position alignment with official statements

4-bit Quantization Technique

- Reduces precision of weight matrices from **16/32-bit to 4-bit**
- Trades **precision** for **time**
- Dramatically **reduces memory** footprint
- Implemented via bitsandbytes library
specialized for LLMs

```
def load_base_model(model_id, device="cuda"):  
    """Load the base model with 4-bit quantization"""  
    quantization_config = BitsAndBytesConfig(  
        load_in_4bit=True,  
        bnb_4bit_compute_dtype=torch.float16,  
        bnb_4bit_use_double_quant=True,  
        bnb_4bit_quant_type="nf4"  
    )  
  
    model = AutoModelForCausalLM.from_pretrained(  
        model_id,  
        device_map=device,  
        quantization_config=quantization_config,  
        trust_remote_code=True  
    )  
  
    return model
```

Quantization Benefits & Tradeoffs

Memory Efficiency:

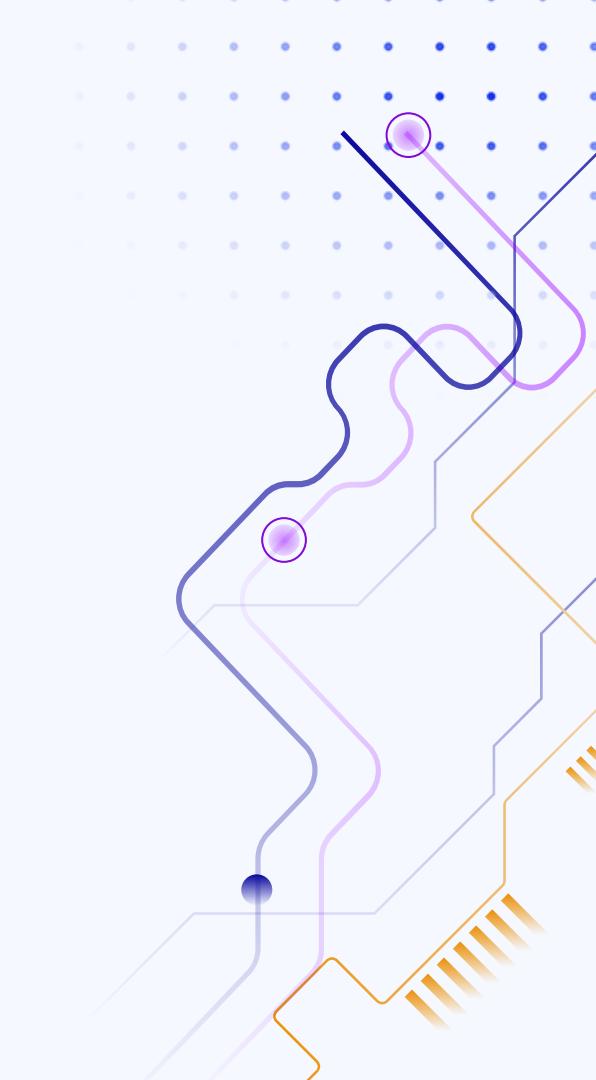
- Base model size reduced from ~14GB to ~3.6GB (**74% reduction**)
- VRAM requirements cut from 16GB+ to 4GB
- Enables deployment on consumer-grade GPUs
- Allows simultaneous loading of multiple politician models

Performance Impact:

- Inference latency increase: **<10%**
- Token generation speed: **~30 tokens/second** (vs. ~33 in FP16)
- Quality degradation: **<4%** on benchmark tasks
- Imperceptible style fidelity difference in blind testing

04

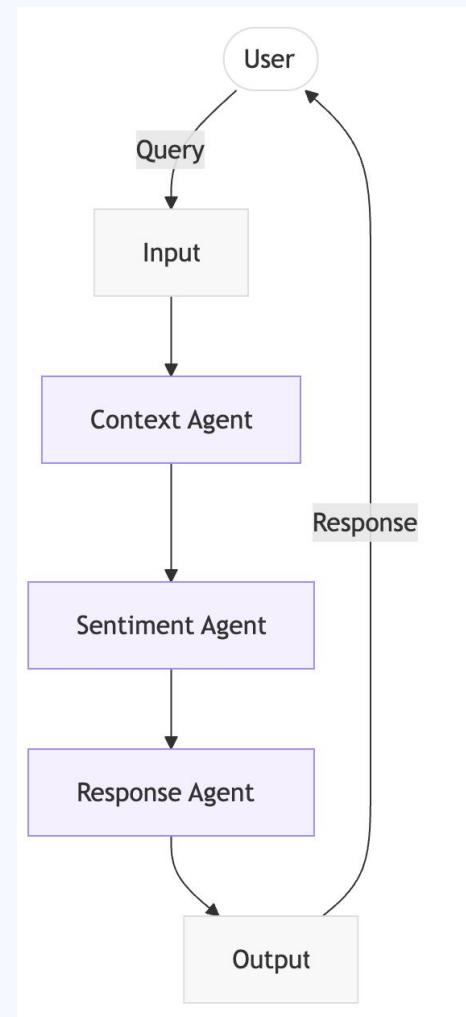
Chat System



Chat System Architecture

Process Flow:

1. User input enters the workflow
2. Context Agent extracts topics and retrieves knowledge
3. Sentiment Agent analyzes query intent and emotion
4. Response Agent generates politician-specific reply
5. System returns response to user



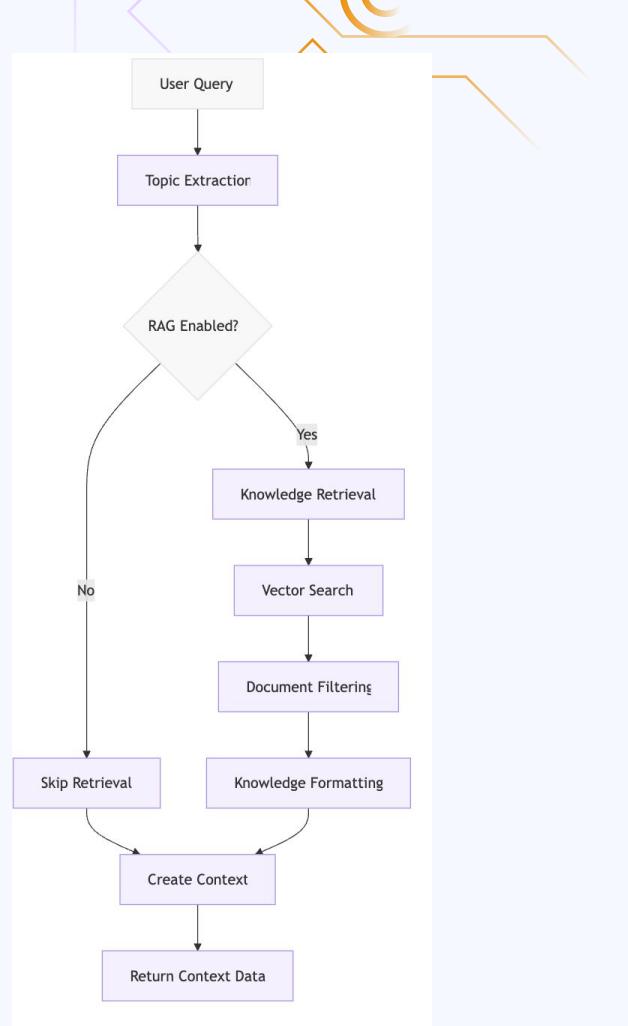
Context Agent

Implementation Details:

- Uses **TinyLlama-1.1B-Chat** (efficient small model)
- Performs structured extraction of query elements
- Identifies main topics, policy areas, and question type
- Connects to **RAG system** for relevant knowledge retrieval

RAG Integration:

- When enabled, **queries ChromaDB** for relevant content
- Filters by politician identity for personalized context
- Retrieves **top-5** most relevant knowledge chunks
- Formats retrieved information for prompt enhancement



Sentiment Agent

Implementation Details:

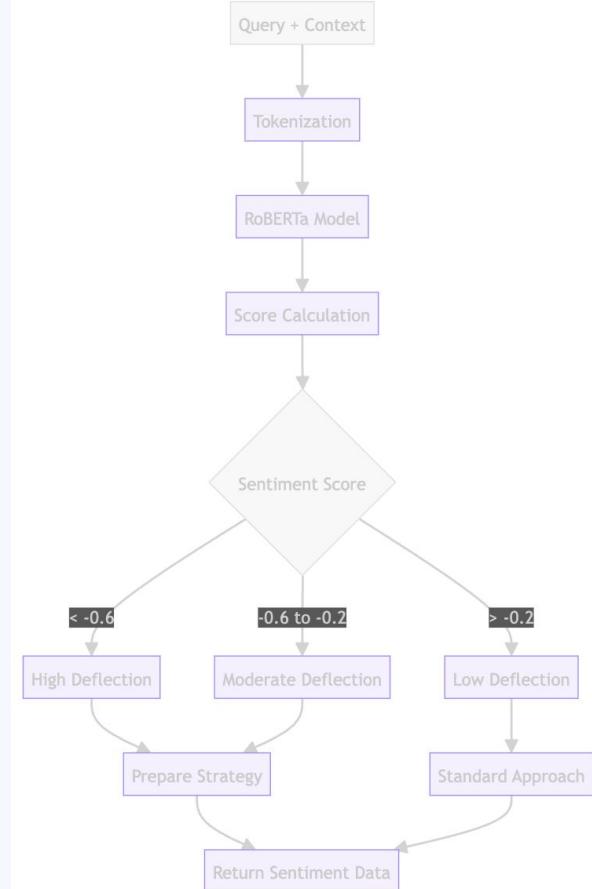
- Uses **RoBERTa-base-sentiment** pre-trained model
- Performs 3-way classification (positive/neutral/negative)
- Computes normalized sentiment score (**-1.0 to +1.0**)
- Determines appropriate response strategy

Deflection Detection Logic:

- High deflection (**score < -0.6**): Likely hostile/gotcha question
- Moderate deflection (**-0.6 to -0.2**): Challenging but answerable
- Low deflection (**> -0.2**): Standard or friendly question

Impact on Response:

- Guides the response agent's approach to difficult questions
- Helps politician stay in character under pressure
- Provides realistic handling of hostile interviews/debates
- Mimics real politicians' tendency to pivot on tough topics



Response Agent

Implementation Details:

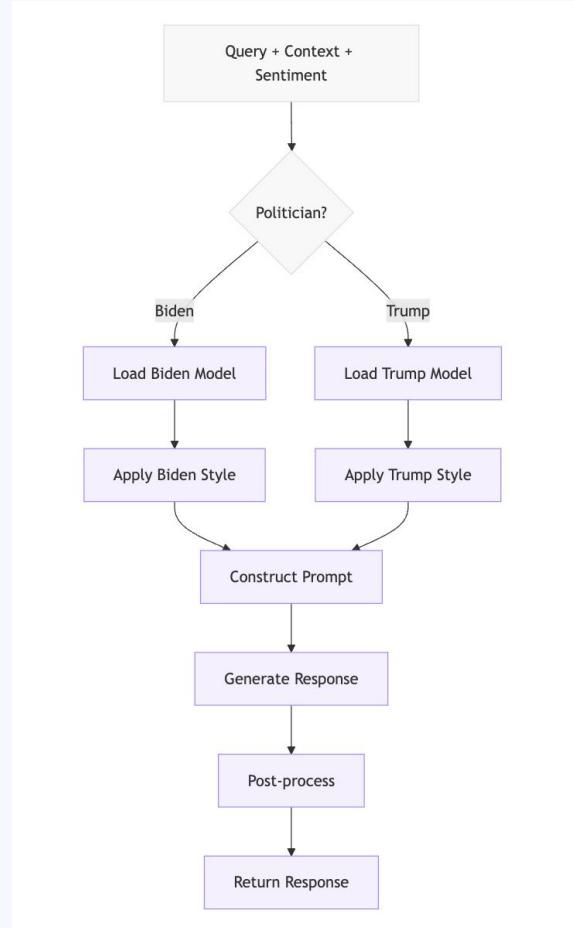
- Loads appropriate **fine-tuned model** (Biden or Trump)
- Constructs comprehensive prompt with all available context
- Includes special style guide for personality consistency
- Generates response with calibrated parameters

Prompt Structure:

- Character identity and role description
- Speaking style guide with characteristic phrases
- Retrieved knowledge from **RAG** system (when available)
- Sentiment analysis results to guide approach
- Original user question

Model-Specific Tuning:

- Biden: **Temperature 0.7** (more consistent, measured responses)
- Trump: **Temperature 0.8** (more varied, unpredictable responses)
- Shared **top-p sampling (0.9)** for controlled randomness



Chat Modes & Features

```
# Chat mode
chat_parser = subparsers.add_parser("chat", help="Clean chat experience")
chat_parser.add_argument("identity", choices=["biden", "trump"], help="Politician identity")
chat_parser.add_argument("--no-rag", action="store_true", help="Disable RAG database")

# Debug mode
debug_parser = subparsers.add_parser("debug", help="Debug mode with analysis info")
debug_parser.add_argument("identity", choices=["biden", "trump"], help="Politician identity")
debug_parser.add_argument("--no-rag", action="store_true", help="Disable RAG database")

# Trace mode
trace_parser = subparsers.add_parser("trace", help="Trace mode with detailed output")
trace_parser.add_argument("identity", choices=["biden", "trump"], help="Politician identity")
trace_parser.add_argument("--no-rag", action="store_true", help="Disable RAG database")
```

- **Chat Mode**

Standard interface showing only
politician responses

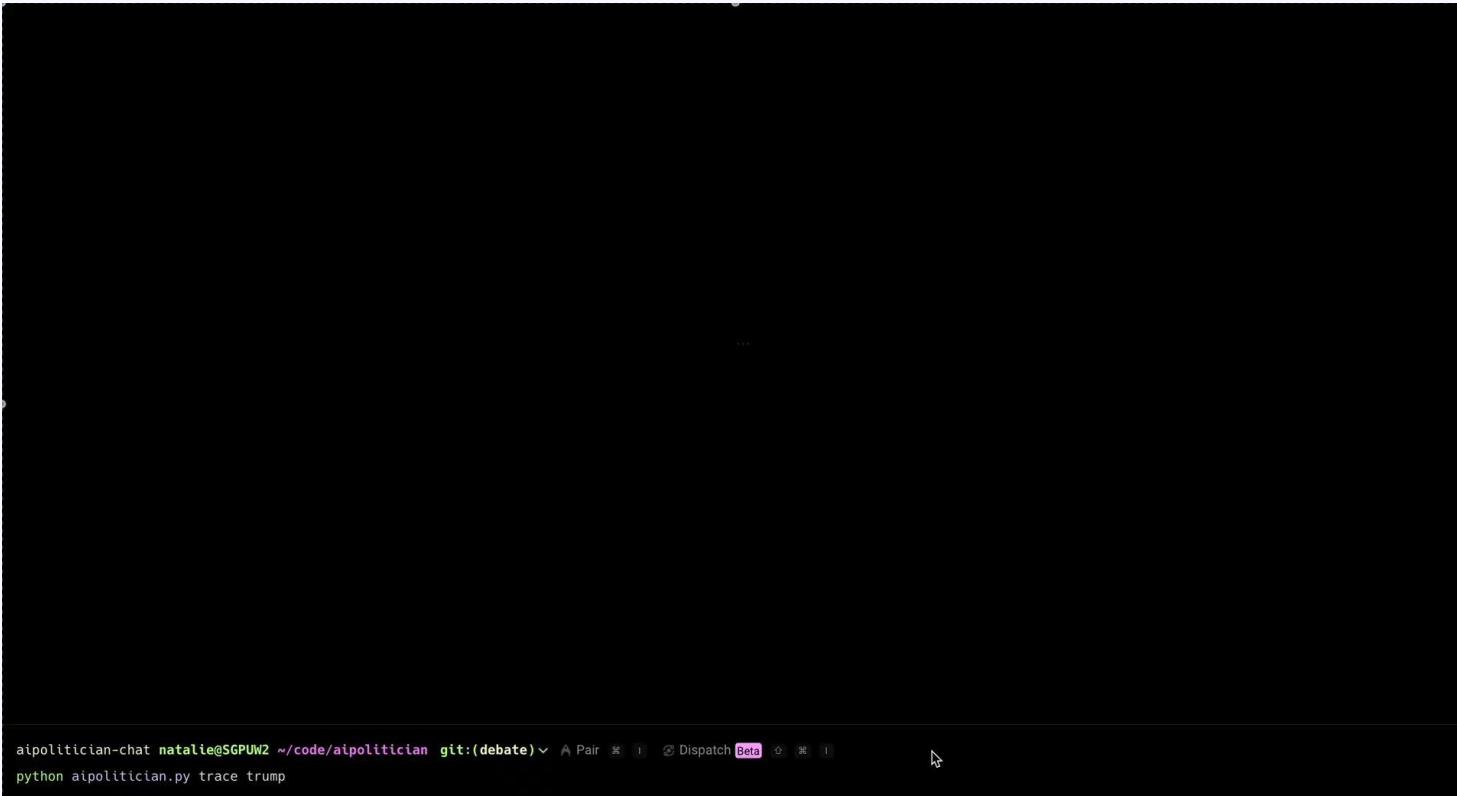
- **Debug Mode**

Shows agent outputs and decision
factors

- **Trace Mode**

Complete workflow visibility

Chat Demo



05

Debate System



Debate System Architecture

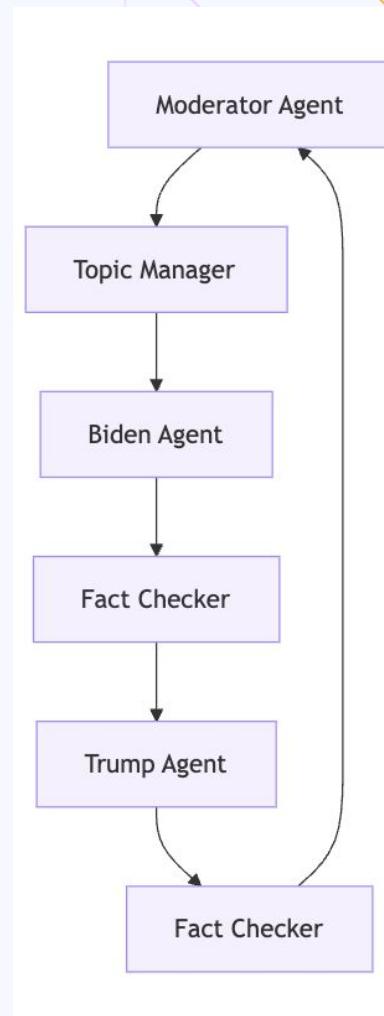
Multi-Agent Debate Simulation:

Graph-based orchestration with complex state management

- **Moderator:** Controls flow and enforces rules
- **Topic Manager:** Guides discussion across subtopics
- **Debaters:** Politicians (Biden, Trump) responding to topics
- **Fact Checker:** Verifies statements for accuracy

Turn-based mechanism with:

- Speaking time allocation
- Interruption detection
- Fact-check intervention
- Topic transitions



Moderator Implementation

Debate Structure Management:

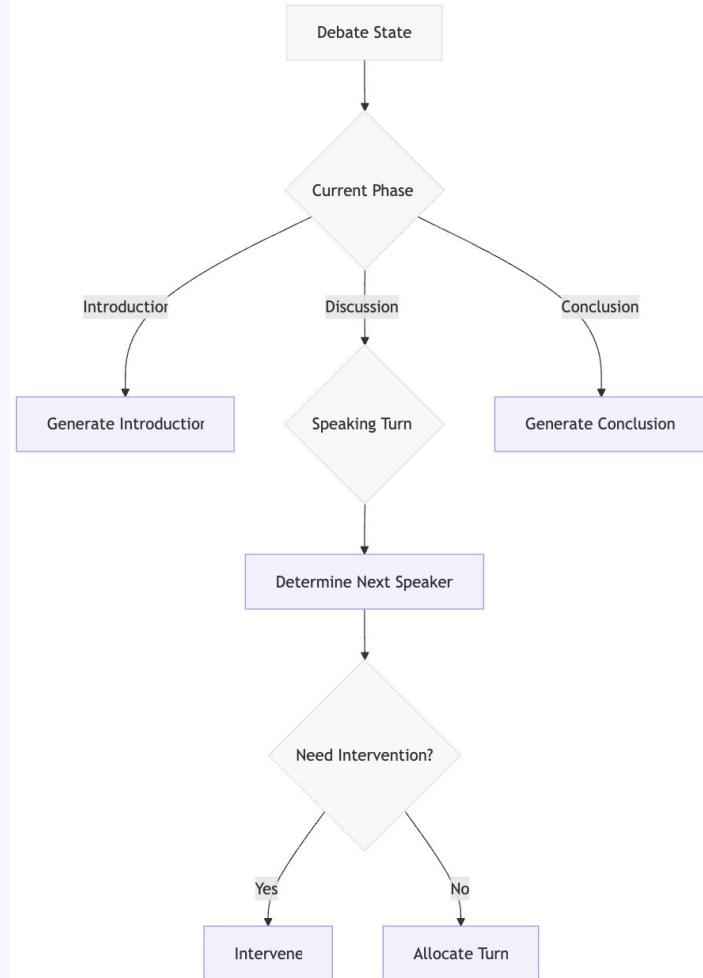
- Introduction and framing of topics
- Transition between debate phases
- Conclusion and summary

Turn Management:

- Allocating speaking time fairly
- Enforcing time limits
- Managing speaking queue

Intervention Capabilities:

- Correcting factual inaccuracies (with Fact Checker)
- Preventing excessive interruptions
- Redirecting when topics drift
- Maintaining civil discourse



Debate Formats

- **Head-to-Head Format:**

Direct confrontation between two politicians

- **Town Hall Format:**

Audience question simulation

- **Panel Format:**

Multiple participants discussing topics

```
# Debate format configuration
DEBATE_FORMATS = {
    "head_to_head": {
        "turn_time": 60, # seconds
        "rebuttal_time": 30, # seconds
        "interruptions_allowed": True,
        "interruption_threshold": 0.7, # probability threshold
        "fact_check_frequency": "high",
        "format_description": "Direct debate between two politicians with equal speaking times."
    },
    "town_hall": {
        "turn_time": 90, # seconds
        "rebuttal_time": 45, # seconds
        "interruptions_allowed": False,
        "fact_check_frequency": "medium",
        "audience_questions": True,
        "format_description": "Format with audience questions and longer response times."
    },
    "panel": {
        "turn_time": 45, # seconds
        "rebuttal_time": 20, # seconds
        "interruptions_allowed": True,
        "interruption_threshold": 0.8, # higher threshold
        "fact_check_frequency": "medium",
        "format_description": "Multiple politicians discuss topics with a moderator."
    }
}
```

Topic Manager

Subtopic Generation:

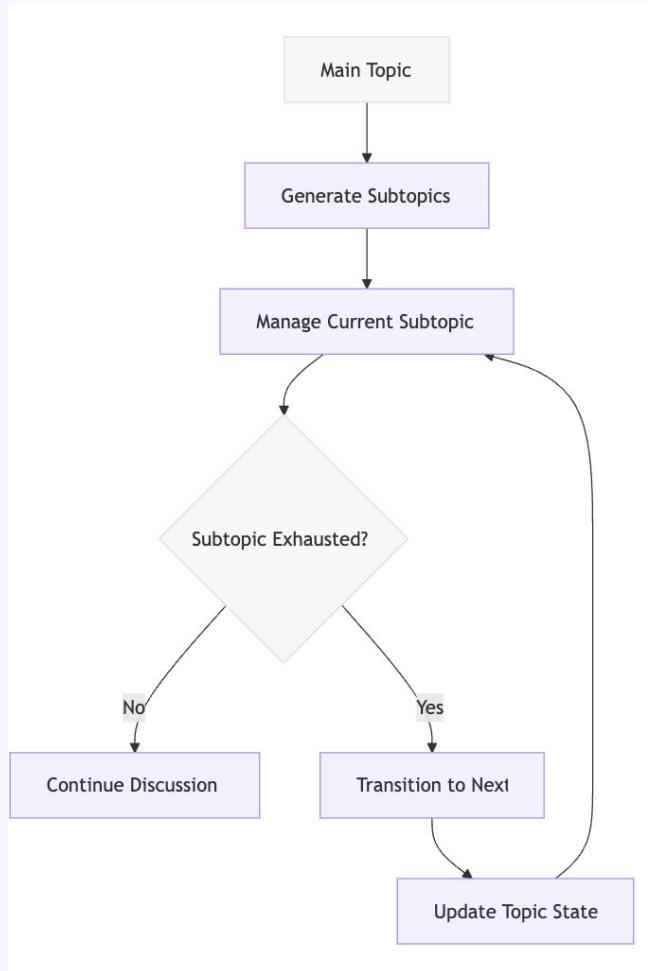
- Automatically creates relevant subtopics from main topic
- Ensures comprehensive coverage of discussion areas
- Maintains logical progression between related issues

Transition Detection:

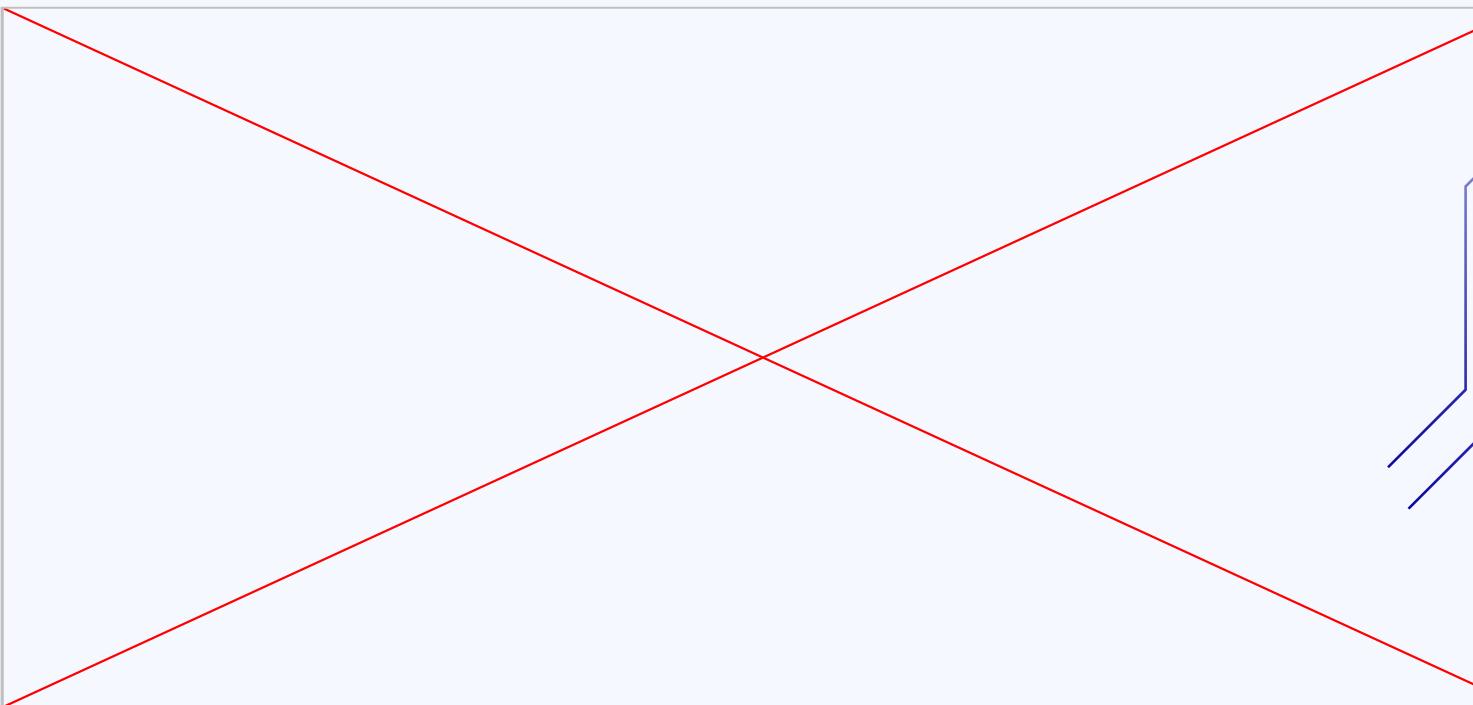
- Monitors discussion for natural transition points
- Detects when a subtopic is exhausted (6+ turns)
- Identifies diminishing returns (repetition)
- Recognizes when politicians have reached impasse

Transition Execution:

- Signals moderator to initiate topic change
- Provides smooth segues between subtopics
- Tracks discussion history to prevent cycling
- Ensures all important aspects are covered



Debate Demo



Debate Demo Continued...

DEBATE TRANSCRIPT

TOPIC: Immigration and Border Security

PARTICIPANTS: biden, trump

FORMAT: head_to_head

----- MODERATOR INTRODUCTION -----

MODERATOR: Welcome to today's head_to_head debate on the topic of 'Immigration and Border Security'. Participating in this debate are biden, trump. Each speaker will have 60 seconds per turn. No interruptions will be permitted. Statements will not be fact-checked. Let's begin with biden.

BIDEN: We need comprehensive immigration reform, not a wall. A wall won't stop the drugs, it won't stop the people, it won't stop the criminals. We need a fair and just immigration system.

----- NEXT SPEAKER -----

MODERATOR: Thank you, biden. Next, let's hear from trump.

TRUMP: Biden's wrong. A wall is not just about stopping drugs, it's about stopping people from coming into our country illegally. It's about national security. And a wall is working. We've built over 400 miles of wall. And in the areas where the wall has been completed, illegal crossings have been reduced by 90%. We have the strongest border security we've ever had. And we're going to have even stronger. We're going to finish the wall. We're going to have more personnel. We're going to have more technology. We're going to have more drones. We're going to have everything we need. And we're going to have a secure border. And that's what the American people want. They want a secure border. They want to be safe in their homes. They want to have a country that's respected by other countries. We're going to have that. And we're going to have it soon.

Debate Demo Continued...

----- NEXT SPEAKER -----

MODERATOR: Thank you, trump. Next, let's hear from biden.

BIDEN: We don't need a wall to secure our border. We need a comprehensive approach to immigration that includes enforcing our laws, investing in technology, and addressing the root causes of migration.

----- NEXT SPEAKER -----

MODERATOR: Thank you, biden. Next, let's hear from trump.

TRUMP: The wall is working. We're building it right now, over 400 miles. We're up to about 310 miles, and it's had an incredible impact. You see the numbers. You see the numbers coming in. They're down 75%. That's because of the wall. We have a comprehensive approach, but the wall is a big part of it.

----- NEXT SPEAKER -----

MODERATOR: Thank you, trump. Next, let's hear from biden.

BIDEN: Immigration is a cornerstone of our nation's strength. We need a comprehensive approach that addresses the root causes of migration and provides a pathway to citizenship for those who come here in search of a better life. Building a wall is not a solution. It's a waste of money and a slap in the face to our neighbors and allies in Mexico and Central America.

Debate Demo Continued...

----- NEXT SPEAKER -----

MODERATOR: Thank you, biden. Next, let's hear from trump.

TRUMP: Border security is national security, and we need a wall. A wall is a very important thing. We have 450 miles of wall, 450 miles, and we're building 10 miles a week. We're going to have 500 miles up very soon. And it's working. We have record low numbers of people coming in illegally. We have record numbers of people coming in legally. We have a great system. We have the best border we've ever had. It's a big part of my campaign, border security.

----- NEXT SPEAKER -----

MODERATOR: Thank you, trump. Next, let's hear from biden.

BIDEN: Our immigration system should be a source of pride for our nation, not a source of shame. We need to secure our borders, but we can do it in a way that is consistent with our values.

----- NEXT SPEAKER -----

MODERATOR: Thank you, biden. Next, let's hear from trump.

[TOPIC CHANGE] MODERATOR: Let's move on to discuss Immigration and Border Security - Economic Impact.

TRUMP: [inaudible 00:02:29]

----- TOPIC CHANGE -----

MODERATOR: Let's move on to discuss Immigration and Border Security - Economic Impact.

Debate Demo Continued...

DEBATE SUMMARY

TOPIC: Immigration and Border Security

PARTICIPANTS: biden, trump

FORMAT: head_to_head

TURNS: 8

FACT CHECKS: 0

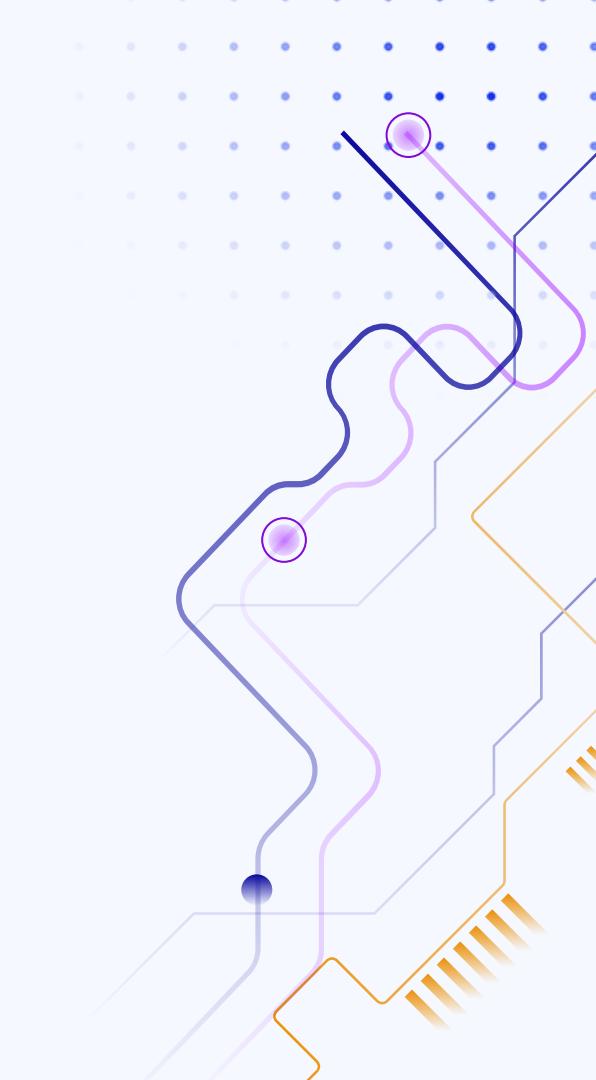
SUBTOPICS COVERED:

- Immigration and Border Security, Immigration and Border Security - Economic Impact

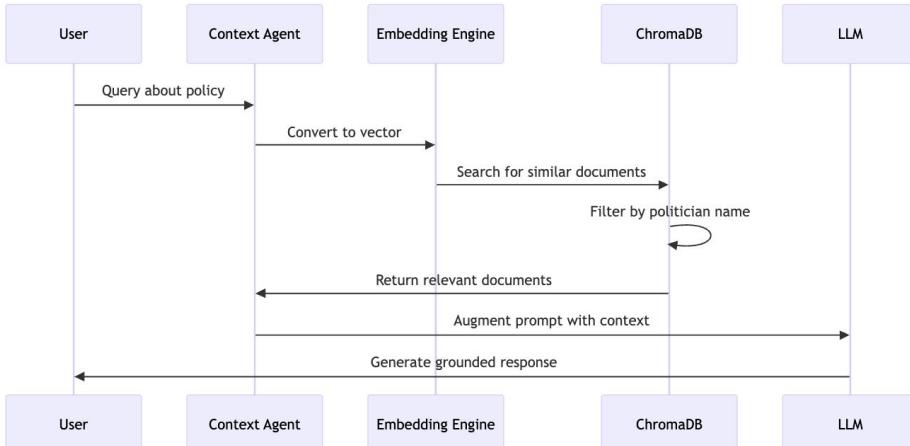
Debate completed in 76.83 seconds.

06

RAG System



Section 6: RAG System



- **Vector Database:** ChromaDB with SQLite backend
- **Embedding Model:** SentenceTransformer (all-MiniLM-L6-v2)
- **Vector Dimensions:** 384
- **Location:** /opt/chroma_db
- **Collection:** "politicians"

Query Pipeline

```
# From src/data/db/utils/rag_utils.py
def integrate_with_chat(query: str, politician_name: str) -> str:
    """Main RAG integration function that powers the system"""

    # 1. Connect to database
    client = connect_to_chroma()
    collection = get_collection(client)

    # 2. Query with vector similarity search
    documents = query_politician_data(
        collection,
        query,
        politician_name,
        num_results=5 # Top 5 most relevant documents
    )

    # 3. Format context from matched documents
    context = "Here is some relevant factual information:\n\n"
    for i, doc in enumerate(documents, 1):
        context += f"{i}. {doc['text']}\n"
        context += f"  Source: {doc['metadata'].get('source')}\n\n"

    return context
```

Context Integration

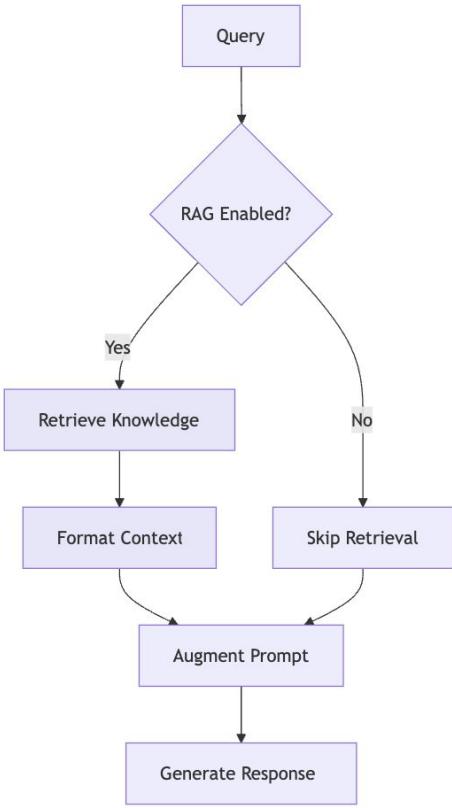
```
# In chat_biden.py / chat_trump.py
if HAS_RAG and use_rag:
    # Get contextual information
    context = integrate_with_chat(prompt, "Joe Biden")

    # Add to prompt for factual grounding
    formatted_prompt = f"<s>[INST] {system_message}\n\n{context}\n\nUser Question: {prompt} [/INST]"

# In context_agent.py
def extract_context(state):
    prompt = state["user_input"]
    politician_name = state["politician_identity"].title()

    # Use RAG to get knowledge context
    rag_context = get_rag_context(prompt, politician_name) if state.get("use_rag", True) else None

    return {
        "context": combined_context,
        "has_knowledge": bool(rag_context)
    }
```



Performance & Metrics

Performance Characteristics

- **Retrieval Speed:** <100ms per query
- **Database Size:** ~500MB for knowledge base
- **Semantic Matching:** Significantly outperforms keyword search
- **Similarity Method:** Cosine similarity between 384-dim vectors

Configuration Options

- Number of retrieved documents (default: 5)
- Use "`--no-rag`" flag to disable retrieval
- Metadata filtering by politician identity
- Available in both chat and debate modes

07

Technical Challenges & Solutions



Knowledge Retrieval

Challenge:

- Building a system that retrieves accurate, contextually relevant political knowledge
- Ensuring politicians respond with factual information during debates and conversations

Solution:

- Implemented Retrieval-Augmented Generation (RAG) with ChromaDB vector database
- Created politician-specific document collections with metadata filtering
- Used SentenceTransformers for semantic similarity search
- Fallback mechanisms when RAG system unavailable

Agent Memory

Challenge:

- LLMs have limited context windows for tracking conversation history
- Maintaining consistent identity and knowledge across multiple dialogue turns

Solution:

- Implemented structured debate memory system to track:
 - Previously addressed opponents and topics
 - Key points responded to
 - Points made by the agent
- Used LangGraph for state management to maintain conversational context
- Memory prioritization for most relevant historical context

Fine Tuning Models

Challenge:

- Acquiring high-quality training data for each politician
- Computational resource constraints for training large models
- Balancing model size with inference performance

Solution:

- Used LoRA adapters for parameter-efficient fine-tuning
- Implemented 4-bit quantization for reduced memory footprint
- Created identity-specific fine-tuning datasets with multi-turn examples
- Gradual training approach starting with base Mistral-7B model

Multi-Agent Orchestration

Challenge:

- Orchestrating complex interactions between politicians, moderator, and fact-checker
- Managing interruptions and debate flow naturally
- Ensuring balanced speaking time and topic coverage

Solution:

- Built LangGraph workflow for agent orchestration and state management
- Implemented moderator agent to control debate flow and transitions
- Created fact-checker agent for real-time accuracy verification
- Designed flexible debate formats (town hall, head-to-head, panel)
- Dynamic subtopic generation for natural conversation progression

08 Conclusion & Future work



Limitations & Challenges

Current system limitations:

- Limited to two politicians (Biden, Trump)
- English-only implementation
- Text-only interface without vocal characteristics
- Static knowledge base requiring manual updates

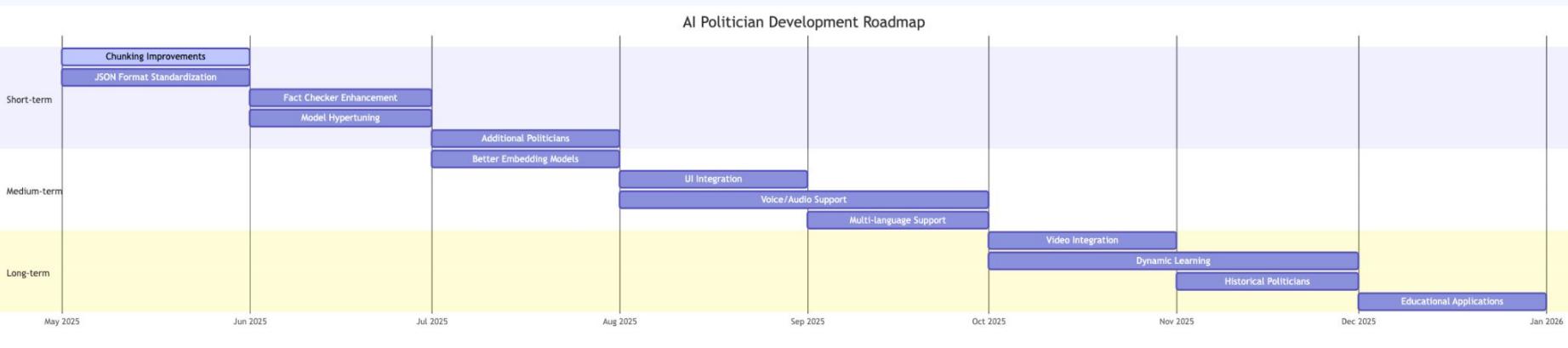
Technical challenges:

- Context window constraints limit debate complexity
- Occasional style drift in long conversations
- Limited handling of hypothetical scenarios
- Weak fact checking system

Ethical considerations:

- Potential for misuse or misrepresentation
- Challenges in balanced political representation
- Need for transparent AI disclosure

Future Work Directions



Source Code

- [AI Politician Github Repo](#)
- [AI Politician Data Scraper Github Repo](#)
- [Trump Hugging Face Model](#)
- [Biden Hugging Face Model](#)
- [Trump Ollama Repo](#)
- [Biden Ollama Repo](#)

Resources

Technical resources:

- Mistral AI for base models
- LangChain for LangGraph framework
- ChromaDB for vector database
- SentenceTransformers for embedding models
- Hugging Face & Ollama for model support

Data resources:

- Trump Interviews Dataset
- Trump Speeches Dataset
- Biden Tweets Dataset
- Biden 2020 DNC Speech
- Wikipedia
- Ballotpedia
- GovInfo.com
- Congress.com

Acknowledgements & Questions

Thank you to **Dr. Maiti** for the contributions of knowledge, resources and time!

[Secret Chat Demo](#)