Abascus ChatLLM Node for n8n

This custom node for n8n enables seamless integration with Large Language Models (LLMs) through the Abascus AI platform, specifically designed for the Luxoranova Executive AI Subbrain's 60-second neural loop system.

Features

- Connect to OpenAI or Abascus AI API endpoints
- Support for multiple LLM models (GPT-4, GPT-3.5, Claude, etc.)
- Task-specific prompting for sales, outreach, content creation, and investor relations
- Contextual memory to maintain conversation history across workflow runs
- Structured data extraction based on task type
- Customizable model parameters (temperature, max tokens, etc.)
- Support for both chat completions and text completions
- JSON response format option for structured outputs

Installation

Method 1: Manual Installation

1. Create a `.n8n/custom` directory in your home directory if it doesn't exist:

   ```
   mkdir -p ~/.n8n/custom
   ```

2. Copy the `abascus_chatllm_node.json` file to this directory:

   ```
   cp ~/luxoranova/abascus_chatllm_node.json ~/.n8n/custom/
   ```

3. Create an SVG icon file named `abascus.svg` in the same directory:

   ```
   # You can use a placeholder SVG or create a custom one
   echo '<svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24
   ```

4. Restart your n8n instance:

   ```
   # If running as a service
   sudo systemctl restart n8n

   # If running via npm
   npm run start
   ```

Method 2: Docker Installation

If you're running n8n in Docker, you need to mount the custom node directory:

1. Create a directory on your host machine:

   ```
   mkdir -p /path/on/host/custom
   ```

2. Copy the node file to this directory:

```
cp ~/luxoranova/abascus_chatllm_node.json /path/on/host/custom/
```

3. Add the SVG icon to the same directory:

```
# Create or copy your SVG icon
cp abascus.svg /path/on/host/custom/
```

4. Update your Docker run command or docker-compose.yml to mount this directory:

```
docker run -it --rm \
  -v /path/on/host/custom:/home/node/.n8n/custom \
  -p 5678:5678 \
  n8nio/n8n
```

Or in docker-compose.yml:

```
version: '3'
services:
  n8n:
    image: n8nio/n8n
    ports:
      - 5678:5678
    volumes:
      - /path/on/host/custom:/home/node/.n8n/custom
```

5. Restart your Docker container.

Configuration

API Credentials

Before using the node, you need to set up API credentials:

1. In the n8n interface, go to **Settings** > **Credentials**
2. Click **New Credential**
3. Select either "Abascus API" or "OpenAI API" depending on your preferred service
4. Enter your API key and save

Node Parameters

The Abascus ChatLLM node offers the following configuration options:

- **Authentication**: Choose between Abascus API or OpenAI API
- **Operation**: Select Chat Completion or Text Completion
- **Model**: Choose from GPT-4, GPT-3.5, Claude, or custom models
- **Task Type**: Specify the type of task (Sales, Outreach, Content Creation, Investor Relations, or General)
- **System Message**: Define the AI assistant's behavior (for chat completion)

- **Use Dynamic System Message**: Automatically use task-specific system messages
- **Messages**: Define the conversation history (for chat completion)
- **Prompt**: Specify the input text (for text completion)
- **Include Previous Context**: Enable contextual memory across workflow runs
- **Context Key**: Specify the key for storing context data
- **Options**: Configure model parameters like temperature, max tokens, etc.

Usage Examples

Basic 60-Second Neural Loop

This example shows a basic implementation of the 60-second neural loop using the Abascus ChatLLM node:

1. Add a **Schedule Trigger** node:
   - Set Interval to "Every Minute"
2. Add a **Function** node to prepare input:

```
// Get current task from a rotation of tasks
const tasks = ['sales', 'outreach', 'content', 'investor'];
const currentHour = new Date().getHours();
const taskIndex = currentHour % tasks.length;
const currentTask = tasks[taskIndex];

// Prepare task-specific input
let input;
switch(currentTask) {
  case 'sales':
    input = "Generate a follow-up email for prospect XYZ who showed interest in our AI
    break;
  case 'outreach':
    input = "Draft a LinkedIn message to connect with potential partner ABC Company";
    break;
  case 'content':
    input = "Create an outline for a blog post about the benefits of AI automation in b
    break;
  case 'investor':
    input = "Prepare a summary of our Q2 performance highlights for investors";
    break;
}

return {
  json: {
    input,
```

```
    taskType: currentTask
  }
};
```

3. Add the **Abascus ChatLLM** node:
   - Set Authentication to your preferred API
   - Set Operation to "Chat Completion"
   - Set Model to "GPT-4" or your preferred model
   - Set Task Type to `={{ $json.taskType }}`
   - Enable "Use Dynamic System Message"
   - Enable "Include Previous Context"
   - Set Context Key to "neuralLoopContext"
   - Configure any additional options as needed
4. Add a **Function** node to process the response:

```
// Extract the AI response
const response = $input.item.json;

// Log the completion of this neural loop cycle
console.log(`Completed ${response.taskType} task at ${response.timestamp}`);

// Process based on task type
switch(response.taskType) {
  case 'sales':
    // Store in sales CRM or trigger email send
    break;
  case 'outreach':
    // Schedule social media post or add to outreach queue
    break;
  case 'content':
    // Add to content calendar or send to review
    break;
  case 'investor':
    // Update investor dashboard or prepare for review
    break;
}

return {
  json: {
    success: true,
    taskCompleted: response.taskType,
    timestamp: response.timestamp,
    content: response.content || response.text,
    structuredData: response.structuredData || {}
  }
};
```

5. Add a **Google Firestore** node to store results:
    - Set Operation to "Create Document"
    - Set Collection to "neuralLoopResults"
    - Set Document ID to `={{ $json.timestamp }}`
    - Set Fields to map to the output from the previous node

Multi-Agent Workflow with Gatekeeper

For more complex scenarios, you can implement a multi-agent system with a gatekeeper pattern:

1. Start with the same **Schedule Trigger** node
2. Add an **HTTP Request** node to fetch pending tasks:
    - Configure to fetch tasks from your API or database
3. Add a **Function** node to prepare the gatekeeper prompt:

```
const tasks = $input.item.json.tasks;

return {
  json: {
    input: `You are the gatekeeper AI. Review these pending tasks and determine which s
    taskType: 'general'
  }
};
```

4. Add the **Abascus ChatLLM** node as the gatekeeper:
    - Configure with appropriate settings
    - Set Response Format to "JSON Object"
5. Add a **Split In Batches** node to process each task assignment
6. Add a **Switch** node to route tasks to different branches based on the assigned specialist
7. Add specialized **Abascus ChatLLM** nodes for each specialist:
    - Configure each with the appropriate task type
    - Use task-specific system messages and parameters
8. Add a **Merge** node to combine all results
9. Add a final **Function** node to process and store all results

Troubleshooting

Node Not Appearing

If the node doesn't appear in your n8n interface after installation:

1. Check that the file is in the correct location
2. Verify that the node file has the correct format and no syntax errors
3. Clear your browser cache
4. Restart n8n completely

API Connection Issues

If you're experiencing connection issues with the API:

1. Verify your API key is correct and has sufficient permissions
2. Check your internet connection
3. Ensure the API endpoint is accessible from your n8n instance
4. Check for any rate limiting or quota issues with your API provider

Context Not Persisting

If the conversation context isn't persisting between workflow runs:

1. Ensure "Include Previous Context" is enabled
2. Verify the Context Key is consistent across workflow runs
3. Check that your n8n instance is properly saving workflow static data
4. Consider increasing the context limit if you need longer conversation history

Support

For issues with this custom node, please contact the Luxoranova support team or open an issue in the project repository.

License

This custom node is provided as part of the Luxoranova Executive AI Subbrain deployment kit and is subject to the terms of the Luxoranova license agreement.