Text Content Corpus Tool

A Node.js-based tool for managing and storing text documents (books, articles, and other content) in a MongoDB corpus database.

Features

- 📚 Store text documents with rich metadata
- 🏷️ Attribution tracking (title, author, content type)
- 📊 Collection statistics and health monitoring
- 🌐 Web interface for document management
- ⚡ RESTful API for programmatic access

Prerequisites

Before running this tool locally, ensure you have:

- **Node.js** (version 14 or higher)
- **MongoDB** (local installation or MongoDB Atlas account)
- **Git** (for cloning if needed)

Installation

1. **Clone or download the project files**

```bash
git clone <your-repo-url>
cd text-corpus-tool
```

2. **Install dependencies**

```bash
npm install
```

3. **Create environment configuration** Create a `.env` file in the root directory:

```env
```

```
# MongoDB Configuration
MONGODB_URI=mongodb://localhost:27017/text_corpus

# Server Configuration
PORT=3000
```

MongoDB Options:

- **Local MongoDB:** `mongodb://localhost:27017/text_corpus`
- **MongoDB Atlas:** `mongodb+srv://username:password@cluster.mongodb.net/text_corpus`

Required Dependencies

Make sure your `package.json` includes these dependencies:

```json
{
  "name": "text-corpus-tool",
  "version": "1.0.0",
  "dependencies": {
    "express": "^4.18.0",
    "cors": "^2.8.5",
    "dotenv": "^16.0.0",
    "mongodb": "^6.0.0"
  }
}
```

Install them with:

```bash
npm install express cors dotenv mongodb
```

Project Structure

```
text-corpus-tool/
├── server.js            # Main server file
├── services/
│   └── mongoservice.js    # MongoDB service class
├── public/
│   ├── index.html        # Web interface
```

```
│   ├── style.css        # Styles
│   └── script.js        # Frontend JavaScript
├── .env                 # Environment variables
├── package.json         # Node.js dependencies
└── README.md            # This file
```

Running the Tool

1. **Start MongoDB** (if running locally)

```bash
# macOS (with Homebrew)
brew services start mongodb-community

# Ubuntu/Debian
sudo systemctl start mongod

# Windows
net start MongoDB
```

2. **Start the application**

```bash
npm start
# or
node server.js
```

3. **Access the tool**
   - Web Interface: http://localhost:3000
   - API Health Check: http://localhost:3000/api/health

API Endpoints

Document Management

- `POST /api/insert-document` - Add a new text document
- `GET /api/documents?limit=10` - List recent documents (metadata only)

System Information

- `GET /api/stats` - Get collection statistics

- `GET /api/health` - Check system health and database connection

Document Format

When inserting documents via API, use this structure:

```json
{
  "content_text": "The full text content of the document...",
  "attribution": {
    "title": "Document Title",
    "author": "Author Name",
    "content_type": "book|article|essay|other",
    "publication_date": "2024-01-01",
    "source_url": "https://example.com/optional"
  },
  "training_metadata": {
    "character_count": 1500,
    "weighting": 1.0,
    "tags": ["fiction", "classic"]
  }
}
```

Environment Variables

Variable	Description	Default
`MONGODB_URI`	MongoDB connection string	`mongodb://localhost:27017/corpora`
`PORT`	Server port	`3000`

Troubleshooting

MongoDB Connection Issues

- Ensure MongoDB is running: `mongosh` (test connection)
- Check firewall settings if using remote MongoDB
- Verify connection string format in `.env`

Port Already in Use

```bash
```

```
# Find process using port 3000
lsof -i :3000

# Kill the process
kill -9 <PID>
```

Missing Dependencies

```bash
# Clear node modules and reinstall
rm -rf node_modules package-lock.json
npm install
```

Development

Adding New Features

1. Create new routes in `server.js`
2. Extend MongoDB service in `services/mongoservice.js`
3. Update the web interface in `public/`

Database Schema

Documents are stored with this structure:

- `document_id`: Unique identifier
- `content_text`: Full text content
- `attribution`: Title, author, type, etc.
- `training_metadata`: Character count, weighting, tags
- `created_at`: Timestamp
- `updated_at`: Timestamp
- `version`: Document version

Security Notes

- This tool is designed for local development
- For production use, add authentication and input validation
- Consider rate limiting for public deployments

- Use environment variables for sensitive configuration

License

[Add your license information here]

Support

For issues or questions:

1. Check the health endpoint: `/api/health`
2. Review console logs for error details
3. Ensure all dependencies are properly installed