# Document Similarity Analysis Tool

Automated analysis tool for detecting repetitive content patterns in text documents (books, articles, reports, academic papers). Uses advanced NLP techniques to group similar paragraphs into clusters and generates concise summaries for each group.

## Features

- **Automated Content Analysis**: Identifies similar paragraphs using sentence embeddings
- **Content Classification**: Categorizes content types (TOC, tables, headers, main content)
- **AI-Powered Summaries**: Generates cluster summaries using OpenAI GPT models
- **Multiple Output Formats**: CSV, DOCX, JSON exports for different use cases
- **Cross-Platform**: Works on Windows and macOS
- **Bilingual Support**: Handles Polish and English content with appropriate language detection

## Quick Start

### Prerequisites

- Python 3.8 or later
- OpenAI API key (optional, for AI summaries)

### Installation

#### Option 1: Clone the repository

1. **Clone the repository and navigate to it**:

```bash
git clone https://github.com/your-username/document-similarity-analysis.git
cd document-similarity-analysis
```

#### Option 2: Use files from USB drive/local folder

If you're using files from a USB drive (e.g., D:), open Terminal/PowerShell and navigate to the drive:

1. **Navigate to your drive/folder**:

```powershell
cd D:\
```

2. **Check what files you have in the directory**:

```powershell
dir *.txt
```

If you see a file named `requirements_txt.txt`, install the dependencies. If not, locate the file with dependencies.

### Install Dependencies

```powershell
pip install -r requirements_txt.txt
```

**Installation should complete with all packages installed:**

- ✅ sentence-transformers (5.1.0) - for document similarity analysis
- ✅ scikit-learn (1.7.1) - machine learning tools
- ✅ numpy (2.3.2) - numerical computations
- ✅ python-docx (1.2.0) - Word file support

- ✅ PyMuPDF (1.26.4) - PDF file support
- ✅ pandas (2.3.2) - data analysis
- ✅ openai (1.106.1) - OpenAI integration
- ✅ python-dotenv (1.1.1) - environment variables
- ✅ torch (2.8.0) - PyTorch for deep learning
- ✅ transformers (4.56.1) - transformer models

### API Key Configuration (optional)

If you don't have or don't configure an API key, the application will work locally.

**Offline mode (without API key):**

- Document similarity analysis ✅
- Embedding generation ✅
- Content clustering ✅
- CSV, JSON, DOCX reports ✅

**What requires API key:**

- Automatic cluster summaries ❌
- AI analyses with analyze_results_v3.py ❌

**To configure API key:**

1. Visit OpenAI website: https://platform.openai.com/api-keys
2. Follow the instructions to obtain your API key

**Option 1: Using .env file (recommended)**

```powershell
notepad "C:\Users\[YourName]\Documents\DocumentAnalysis\.env"
```

Paste:

```
OPENAI_API_KEY=your_actual_key_here
```

Save file (Ctrl+S) and close.

**Option 2: Using environment variable**

```powershell
setx OPENAI_API_KEY "your_actual_key_here"
```

Then restart PowerShell and the application.

### Basic Usage

1. **Place your document** in the project directory, e.g.: `C:\Users\[YourName]\Documents\DocumentAnalysis\Projects\[ProjectName]\`
2. **Run the analysis**:
   ```bash
   python run_document.py --project my_analysis --file "your_document.pdf"
   ```
3. **Review results** in the automatically created project folder, e.g.: `C:\Users\[YourName]\Documents\DocumentAnalysis\Projects\my_analysis\`

### Test the Installation

Use the provided test files to verify everything works:

```bash
# Test with English document
python run_document.py --project test_en --file "What is Lorem Ipsum.pdf"

# Test with Polish document
python run_document.py --project test_pl --file "Czym jest Lorem Ipsum.pdf"
```

### Complete Workflow

```bash
# Step 1: Main analysis
python run_document.py --project my_thesis --file thesis.pdf

# Step 2: Content classification
python analyze_results_v2.py --project my_thesis

# Step 3: AI summaries (requires API key)
python analyze_results_v3.py

# Step 4: Extract clusters
python extract_clusters.py

# Step 5: Get representative examples
python extract_examples.py --project my_thesis
```

### Output Files

| File | Description |
|---|---|
| `results_doc.csv` | Raw analysis data with paragraph-cluster mapping |
| `[filename]_analysis.docx` | Human-readable analysis report |
| `[filename]_analysis.json` | Machine-readable data export |
| `analysis_report.txt` | Summary statistics and insights |
| `clusters/` | Individual cluster files |

## Configuration Options

### Similarity Sensitivity

- `--eps 0.3` (default) - Standard sensitivity
- `--eps 0.2` - Strict matching (fewer, more precise clusters)
- `--eps 0.4` - Loose matching (more clusters, broader similarity)

### Minimum Cluster Size

- `--min-samples 2` (default) - Include all duplicates
- `--min-samples 3` - Only clusters with 3+ paragraphs

### Model Selection

- `--model gpt-4o` (default) - Best quality, especially for Polish
- `--model gpt-4o-mini` - Budget-friendly option

## Supported File Formats

- **PDF** (.pdf) - Text extraction with paragraph detection
- **Word** (.docx) - Native paragraph extraction
- **Text** (.txt) - Paragraph detection via double line breaks

## Understanding Results

### Similarity Categories

- **<10%**: Low similarity - mostly unique content
- **10-25%**: Some similarity - related topics
- **25-50%**: Similar content - significant overlap
- **50-75%**: Very similar - likely repetitive
- **>75%**: Nearly identical - strong duplication

### Content Types

- **UNIQUE**: Standalone paragraphs with no duplicates
- **SIMILAR-XX**: Groups of related/duplicate content
- **TOC**: Table of contents entries
- **Header_Footer**: Page numbers, headers, footers
- **Bibliography**: References and citations
- **Table_Figure**: Tables, figures, case studies

## Troubleshooting

### Common Issues

**"No content found"**

- Check if document contains readable text
- Try different file format (PDF → DOCX)
- Ensure document isn't image-based

**"Import error"**

- Install missing dependencies: `pip install -r requirements_txt.txt`
- Check Python version: `python --version` (need 3.8+)

**"No clusters found"**

- Document may have very unique content (good!)
- Try higher eps value: `--eps 0.4`
- Check minimum samples: `--min-samples 2`

**API errors**

- Verify API key is set correctly
- Check OpenAI account credits
- Tool works without API key (no summaries)

### Performance Tips

- For large documents (>100 pages), consider splitting by chapters
- Use `gpt-4o-mini` for initial testing to reduce costs
- Cache embeddings are stored automatically for repeated analysis

## Project Structure

```
your_project/
├── src/                # Source code (if applicable)
├── test_files/         # Sample documents for testing
├── docs/               # Documentation
├── requirements_txt.txt  # Python dependencies
└── README.md           # This file
```

Analysis creates project folders in:

- **Windows**: `C:\Users\[Name]\Documents\DocumentAnalysis\Projects\`
- **macOS**: `~/Documents/DocumentAnalysis/Projects/`

## Use Cases

- **Academic Papers**: Find repetitive explanations or definitions
- **Technical Documentation**: Detect duplicate instructions
- **Business Reports**: Identify redundant analysis sections
- **Books & Articles**: Locate repeated themes or examples
- **Legal Documents**: Find duplicate clauses or terms

## Additional Resources

### Configuration for Windows Users

**Windows Environment Setup:**

```cmd
# Set API key
setx OPENAI_API_KEY "sk-your-key-here"

# Check installation
python --version
pip list | findstr "openai\|sentence"
```

**macOS Environment Setup:**

```bash
# Add to ~/.zshrc or ~/.bash_profile
echo 'export OPENAI_API_KEY="sk-your-key-here"' >> ~/.zshrc

# Check installation
python3 --version
pip3 list | grep -E "openai|sentence"
```

### Example Projects

**Thesis Analysis:**

```bash
python run_document.py --project master_thesis --file "thesis.pdf" --eps 0.25
python analyze_results_v2.py --project master_thesis
python analyze_results_v3.py
```

**Research Paper Analysis:**

```bash
python run_document.py --project paper_2024 --file "paper.docx" --eps 0.3
python extract_examples.py --project paper_2024 --examples 5
```

### Optimal Similarity Thresholds

**Recommended Settings:**

- **Academic papers**: eps=0.25 (strict detection)
- **Business reports**: eps=0.35 (moderate detection)
- **Articles**: eps=0.4 (broad thematic detection)

**Common Patterns in Documents:**

- Repeating definitions in academic work
- Duplicate methodology descriptions
- Similar chapter introductions
- Repeated conclusions and recommendations

## Contributing

1. Fork the repository
2. Create your feature branch (`git checkout -b feature/AmazingFeature`)
3. Commit your changes (`git commit -m 'Add some AmazingFeature'`)
4. Push to the branch (`git push origin feature/AmazingFeature`)
5. Open a Pull Request

## License



## Support

- **Documentation**: See `docs/business_case_v6_pl.md` for detailed Polish documentation
- **Issues**: Create an issue on GitHub for bugs or feature requests
- **Questions**: Use GitHub Discussions for usage questions

## Acknowledgments

- Built with Sentence Transformers for semantic similarity
- Uses OpenAI GPT models for intelligent summarization
- DBSCAN clustering via scikit-learn