RSS Collector

An offline-friendly Google News RSS ingestion pipeline designed for LLM integration. This tool fetches Google News RSS feeds based on configurable keywords, deduplicates articles intelligently, and stores them in a structured format suitable for downstream processing.

🚀 Features

- **Keyword-based RSS fetching** from Google News
- **Intelligent deduplication** based on URL and content hashing
- **Scheduled execution** with configurable timing
- **Offline compatibility** with single-port external communication
- **Robust error handling** with retry mechanisms
- **LLM-ready JSON output** with clean, structured data
- **Comprehensive logging** for monitoring and debugging

📋 Requirements

- Python 3.10 or higher
- RHEL-compatible environment (or any Linux distribution)
- Internet access through a single configurable port
- Optional: Proxy server (Tinyproxy or Squid)

🛠 Installation

Option 1: Using pip

```bash
# Clone the repository
git clone <repository-url>
cd rss_collector

# Install the package
pip install -e .

# For development with testing dependencies
pip install -e ".[dev]"
```

Option 2: Manual setup

```bash
# Install dependencies
pip install -r requirements.txt

# Create necessary directories
mkdir -p feeds logs config

# Copy configuration files
cp config/feeds.json.example config/feeds.json
cp config/settings.json.example config/settings.json
```

⚙️ Configuration

1. Keywords Configuration (`config/feeds.json`)

```json
{
  "keywords": [
    "artificial intelligence",
    "python programming",
    "technology news"
  ],
  "keyword_groups": [
    {
      "name": "Programming",
      "keywords": ["python", "javascript", "software development"]
    },
    {
      "name": "AI/ML",
      "keywords": ["machine learning", "deep learning"]
    }
  ]
}
```

2. System Settings (`config/settings.json`)

```json
{
  "http": {
    "timeout": 30,
    "max_retries": 3,
    "retry_delay": 5,
    "user_agent": "RSS-Collector/1.0"
  },
  "proxy": {
    "enabled": true,
    "host": "localhost",
    "port": 8081,
    "protocol": "http"
  },
  "schedule": [
    {
      "time": "05:00",
      "description": "Morning news fetch"
    },
    {
      "time": "14:00",
      "description": "Afternoon news fetch"
    }
  ],
  "storage": {
    "base_dir": "./",
    "feeds_dir": "feeds",
    "logs_dir": "logs",
    "cleanup_days": 30
  },
  "processing": {
    "group_delay_minutes": 5,
    "max_articles_per_feed": 100
  }
}
```

3. Environment Variables (`.env`)

```bash
# Proxy configuration
HTTP_PROXY=http://localhost:8081
HTTPS_PROXY=http://localhost:8081

# Logging level
LOG_LEVEL=INFO

# Override settings
RSS_BASE_DIR=/custom/path
RSS_MAX_RETRIES=5
```

🚀 Usage

Running Once

```bash
# Run the collector once
python run.py

# Or using the installed command
rss-collector-run
```

Scheduled Execution

```bash
# Start the scheduler (runs continuously)
python -m src.main --scheduled

# Or using the entry point
rss-collector --scheduled
```

Command Line Options

```bash
# Show help
python run.py --help

# Run with custom config
python run.py --config /path/to/config

# Run specific keywords only
python run.py --keywords "python,ai,technology"

# Verbose logging
python run.py --verbose

# Dry run (no actual fetching)
python run.py --dry-run
```

📁 Output Structure

The collector creates the following directory structure:

```
rss_collector/
├── feeds/
│   ├── 2024-01-15.json        # Daily feeds
│   ├── 2024-01-16.json
│   └── ...
├── logs/
│   ├── 2024-01-15.log         # Daily logs
│   ├── 2024-01-16.log
│   └── ...
└── config/
    ├── feeds.json             # Keywords config
    └── settings.json          # System settings
```

JSON Output Format

```json
[
  {
    "fetched_at": "2024-01-15T08:00:00Z",
    "query": "artificial intelligence",
    "source_url": "https://news.google.com/rss/search?q=artificial%20intelligence&hl=e
    "articles": [
      {
        "title": "AI Breakthrough in Natural Language Processing",
        "link": "https://example.com/ai-breakthrough",
        "published": "Mon, 15 Jan 2024 07:30:00 GMT",
        "source": "TechNews",
        "snippet": "Researchers achieve new milestone in AI language understanding..."
      }
    ]
  }
]
```

🔧 Proxy Setup

Using Tinyproxy

```bash
# Install tinyproxy
sudo yum install tinyproxy

# Configure /etc/tinyproxy/tinyproxy.conf
Port 8081
Listen 127.0.0.1
Allow 127.0.0.1

# Start service
sudo systemctl start tinyproxy
sudo systemctl enable tinyproxy
```

Using iptables (Alternative)

```bash
# Route traffic through specific port
sudo iptables -t nat -A OUTPUT -p tcp --dport 80 -j REDIRECT --to-port 8081
sudo iptables -t nat -A OUTPUT -p tcp --dport 443 -j REDIRECT --to-port 8081
```

🧪 Testing

```bash
# Run all tests
python -m pytest

# Run with coverage
python -m pytest --cov=src

# Run specific test file
python -m pytest tests/test_rss_fetcher.py

# Run with verbose output
python -m pytest -v
```

📝 Logging

The application provides comprehensive logging:

- **INFO**: Normal operations, fetch statistics
- **WARNING**: Recovered errors, configuration issues
- **ERROR**: Failed requests, parsing errors
- **DEBUG**: Detailed execution information

Log files are created daily in the `logs/` directory with automatic rotation.

🔍 Monitoring

Check Status

bash

```bash
# View recent logs
tail -f logs/$(date +%Y-%m-%d).log

# Check feed files
ls -la feeds/

# View statistics
python -c "
import json
from datetime import datetime
date = datetime.now().strftime('%Y-%m-%d')
with open(f'feeds/{date}.json') as f:
    data = json.load(f)
    print(f'Total feeds: {len(data)}')
    print(f'Total articles: {sum(len(feed[\"articles\"]) for feed in data)}')
"
```

Health Check Script

bash

```bash
#!/bin/bash
# health_check.sh

LOG_FILE="logs/$(date +%Y-%m-%d).log"
FEED_FILE="feeds/$(date +%Y-%m-%d).json"

if [ -f "$LOG_FILE" ] && [ -f "$FEED_FILE" ]; then
    echo "RSS Collector: OK"
    echo "Last run: $(tail -1 $LOG_FILE | cut -d' ' -f1-2)"
    echo "Articles today: $(jq '[.[].articles | length] | add' $FEED_FILE)"
else
    echo "RSS Collector: ERROR - Missing files"
    exit 1
fi
```

🐛 Troubleshooting

Common Issues

1. **Connection Errors**
    - Check proxy configuration
    - Verify firewall settings
    - Test internet connectivity

2. **Parse Errors**
    - Check RSS feed validity
    - Verify feedparser version
    - Review error logs
3. **Storage Issues**
    - Check disk space
    - Verify directory permissions
    - Review file system errors
4. **Scheduling Problems**
    - Check system timezone
    - Verify cron permissions
    - Review scheduler logs

Debug Mode

```bash
# Enable debug logging
export LOG_LEVEL=DEBUG
python run.py --verbose

# Check configuration
python -c "
from src.config_manager import ConfigManager
config = ConfigManager()
print('Config loaded successfully')
print(f'Keywords: {len(config.keywords)}')
print(f'Schedule: {len(config.schedule)}')
"
```

📊 Performance

Typical Performance Metrics

- **Fetch time**: 2-5 seconds per keyword
- **Parse time**: <1 second per RSS feed
- **Memory usage**: 50-100 MB during execution
- **Storage**: ~1-2 MB per 100 articles

Optimization Tips

1. Adjust `group_delay_minutes` for rate limiting

2. Set `max_articles_per_feed` to control memory usage
3. Use `cleanup_days` to manage disk space
4. Monitor proxy performance for bottlenecks

🤝 Contributing

1. Fork the repository
2. Create a feature branch
3. Make your changes
4. Add tests for new functionality
5. Run the test suite
6. Submit a pull request

Development Setup

```bash
# Clone and setup development environment
git clone <repository-url>
cd rss_collector
pip install -e ".[dev]"

# Pre-commit hooks
pre-commit install

# Run tests
pytest

# Code formatting
black src/ tests/
flake8 src/ tests/
```

📄 License

🆘 Support

For issues and questions:

1. Check the troubleshooting section
2. Review existing issues
3. Create a new issue with detailed information

🔄 Changelog

Version 1.0.0

- Initial release
- Google News RSS fetching
- Intelligent deduplication
- Scheduled execution
- Comprehensive logging
- LLM-ready output format

Note: This tool is designed for educational and research purposes. Ensure compliance with Google's Terms of Service and robots.txt when using this collector.