Sage Oracle Agent Specification

Executive Summary

The Sage Oracle Agent is a sophisticated document processing and knowledge extraction system designed to handle large-scale document conversion, chunking, embedding, and storage operations. Built on the Agency Swarm framework, it leverages Docling's advanced document understanding capabilities to process various document formats (PDFs, HTML, DOCX, etc.) from multiple sources including URLs, sitemaps, and local folders.

Agent Overview

Purpose

The Sage Oracle Agent serves as a comprehensive document processing pipeline that converts documents to markdown, extracts and annotates images, chunks content for optimal retrieval, creates vector embeddings, and stores results in PostgreSQL vector databases. It's designed for enterprise-scale document processing with concurrent execution capabilities.

Target Market

- Enterprise knowledge management systems
- Research institutions processing large document collections
- Content management platforms requiring intelligent document processing
- RAG (Retrieval Augmented Generation) application developers
- Organizations needing automated document conversion and indexing

Value Proposition

- **Scalable Processing**: Handle hundreds of documents concurrently with controlled resource usage
- **Multi-Format Support**: Process PDFs, HTML, DOCX, XLSX, PPTX, and images seamlessly
- **Intelligent Extraction**: AI-powered image annotation and document structure recognition
- **Production Ready**: Robust error handling, progress tracking, and fault tolerance
- **Flexible Input Sources**: Support for URLs, sitemaps, local folders, and mixed sources

Architecture Overview

Core Components

*1. **Pipeline Architecture***

```

Input Sources → Document Pipeline → Storage Systems
↓ ↓ ↓
• URLs - Conversion Stage - PostgreSQL
• Sitemaps - Chunking Stage - File System
• Folders - Embedding Stage - Vector Database
• Mixed - Storage Stage
```

*2. **Concurrency Management***

• **Controlled Concurrency**: Configurable concurrent pipeline execution (default: 10)
• **Resource Management**: Semaphore-based limiting to prevent resource exhaustion
• **Error Isolation**: Individual document failures don't affect batch processing
• **Progress Tracking**: Real-time monitoring of pipeline execution

*3. **Input Source Abstraction***

• **BaseInputSource**: Abstract interface for different input types
• **FolderInputSource**: Process documents from local folders
• **URLInputSource**: Process documents from URLs and sitemaps
• **MixedInputSource**: Combine multiple input sources in single batch

Agent Specifications

Agent Name: Sage Oracle

Role within the Agency

The Sage Oracle Agent serves as the primary document processing and knowledge extraction specialist within the agency. It handles the complete document lifecycle from ingestion to storage, providing other agents with processed, searchable, and embeddable document content.

Core Responsibilities

1. **Document Conversion**: Convert various document formats to structured markdown
2. **Image Processing**: Extract, annotate, and manage document images
3. **Content Chunking**: Intelligently segment documents for optimal retrieval
4. **Vector Embedding**: Create embeddings for semantic search capabilities
5. **Data Storage**: Store processed content in PostgreSQL and file systems

6. **Batch Processing**: Handle large-scale document processing operations
7. **Progress Monitoring**: Track and report processing status and results

Tools Specification

1. ProcessFolderPipeline

• **Description**: Process all documents in a specified folder with concurrent execution
• **Inputs**:
• `folder_path` (str) - Path to folder containing documents
• `file_patterns` (List[str]) - File patterns to process (default: ["*.pdf", "*.html", "*.docx"])
• `max_concurrent` (int) - Number of concurrent pipelines (default: 10)
• `annotate_images` (bool) - Enable AI image annotation (default: True)
• `save_images_as_files` (bool) - Save images as external files (default: True)
• **Validation**:
• Folder path must exist and be accessible
• File patterns must be valid glob patterns
• Max concurrent must be between 1 and 50
• **Core Functions**:
• Scan folder for matching documents
• Create concurrent processing pipelines
• Execute complete document processing workflow
• Track progress and handle errors
• **APIs**: Docling DocumentConverter, OpenAI API (for image annotation)
• **Output**: JSON object with processing results, statistics, and file paths

2. ProcessSitemapPipeline

• **Description**: Process all documents from a website sitemap with full pipeline execution
• **Inputs**:
• `sitemap_url` (str) - URL of the website sitemap
• `max_concurrent` (int) - Number of concurrent pipelines (default: 10)
• `enable_chunking` (bool) - Enable document chunking (default: True)
• `enable_embeddings` (bool) - Enable vector embeddings (default: True)
• `annotate_images` (bool) - Enable AI image annotation (default: True)
• **Validation**:
• Sitemap URL must be accessible
• Max concurrent must be between 1 and 20 (web processing limit)
• **Core Functions**:
• Extract URLs from sitemap
• Create concurrent processing pipelines
• Execute complete document processing workflow
• Store results in PostgreSQL and file system
• **APIs**: Sitemap parsing, Docling DocumentConverter, PostgreSQL, OpenAI API

• **Output**: JSON object with processing results, statistics, and database records

3. ProcessMixedSources

• **Description**: Process documents from multiple sources (folders, URLs, sitemaps) in a single operation
• **Inputs**:
• `folder_paths` (List[str]) - List of folder paths to process
• `urls` (List[str]) - List of URLs to process
• `sitemap_urls` (List[str]) - List of sitemap URLs to process
• `max_concurrent` (int) - Number of concurrent pipelines (default: 10)
• `output_format` (str) - Output format preference (default: "markdown")
• **Validation**:
• At least one input source must be provided
• All paths and URLs must be valid
• Max concurrent must be between 1 and 50
• **Core Functions**:
• Combine multiple input sources
• Create unified processing pipeline
• Execute concurrent processing
• Aggregate results from all sources
• **APIs**: Docling DocumentConverter, PostgreSQL, OpenAI API
• **Output**: JSON object with aggregated processing results and statistics

4. ConvertSingleDocument

• **Description**: Process a single document through the complete pipeline
• **Inputs**:
• `input_source` (str) - File path, URL, or base64 content
• `input_type` (str) - Type: 'file', 'url', 'base64', or 'auto'
• `annotate_images` (bool) - Enable AI image annotation (default: True)
• `save_images_as_files` (bool) - Save images as external files (default: True)
• `enable_chunking` (bool) - Enable document chunking (default: True)
• `enable_embeddings` (bool) - Enable vector embeddings (default: True)
• **Validation**:
• Input source must be valid and accessible
• Input type must be one of the specified options
• **Core Functions**:
• Auto-detect input type if not specified
• Execute complete document processing pipeline
• Store results in appropriate systems
• Return detailed processing results
• **APIs**: Docling DocumentConverter, PostgreSQL, OpenAI API
• **Output**: JSON object with processing results, file paths, and database records

5. BatchProcessDocuments

• **Description**: Process multiple documents with advanced batch management
• **Inputs**:
• `input_sources` (List[str]) - List of file paths, URLs, or base64 content
• `max_workers` (int) - Number of concurrent threads (default: 4)
• `batch_size` (int) - Number of documents per batch (default: 10)
• `retry_failed` (bool) - Retry failed documents (default: True)
• `progress_callback` (str) - Progress tracking method (default: "console")
• **Validation**:
• Input sources list must not be empty
• Max workers must be between 1 and 20
• Batch size must be between 1 and 100
• **Core Functions**:
• Manage concurrent document processing
• Implement retry logic for failed documents
• Track progress and provide updates
• Aggregate results and statistics
• **APIs**: Docling DocumentConverter, PostgreSQL, OpenAI API
• **Output**: JSON object with batch processing results and detailed statistics

6. MonitorProcessingStatus

• **Description**: Monitor and report on active document processing operations
• **Inputs**:
• `operation_id` (str) - ID of the processing operation to monitor
• `include_details` (bool) - Include detailed progress information (default: False)
• `refresh_interval` (int) - Refresh interval in seconds (default: 5)
• **Validation**:
• Operation ID must be valid and exist
• Refresh interval must be between 1 and 60 seconds
• **Core Functions**:
• Track active processing operations
• Monitor progress and performance metrics
• Identify bottlenecks and issues
• Generate status reports
• **APIs**: Internal monitoring system, PostgreSQL
• **Output**: JSON object with current status, progress, and performance metrics

7. ConfigureProcessingPipeline

• **Description**: Configure and customize the document processing pipeline
• **Inputs**:
• `pipeline_config` (dict) - Pipeline configuration parameters
• `stage_settings` (dict) - Individual stage configuration

• `concurrency_settings` (dict) - Concurrency and resource settings
• `output_settings` (dict) - Output format and storage settings
• **Validation**:
• Configuration must be valid JSON
• All required parameters must be present
• Settings must be within acceptable ranges
• **Core Functions**:
• Validate configuration parameters
• Apply pipeline configuration
• Update processing settings
• Test configuration with sample documents
• **APIs**: Internal configuration system
• **Output**: JSON object with configuration status and validation results

8. ExportProcessingResults

• **Description**: Export processed document results in various formats
• **Inputs**:
• `operation_id` (str) - ID of the processing operation
• `export_format` (str) - Export format: 'json', 'csv', 'markdown', 'html'
• `include_embeddings` (bool) - Include vector embeddings (default: False)
• `include_images` (bool) - Include image references (default: True)
• `output_path` (str) - Path for exported files
• **Validation**:
• Operation ID must be valid and exist
• Export format must be supported
• Output path must be writable
• **Core Functions**:
• Retrieve processing results from database
• Format data according to export format
• Generate export files
• Provide download links or file paths
• **APIs**: PostgreSQL, File system
• **Output**: JSON object with export status and file information

Technical Implementation

Pipeline Stages

1. Conversion Stage

• **Purpose**: Convert documents to markdown with image extraction
• **Technology**: Docling DocumentConverter

• **Features**:
• Multi-format support (PDF, HTML, DOCX, etc.)
• AI-powered image annotation
• Layout preservation
• Metadata extraction

2. Chunking Stage*

• **Purpose**: Intelligently segment documents for optimal retrieval
• **Technology**: Docling HybridChunker
• **Features**:
• Hierarchical chunking
• Token-aware segmentation
• Context preservation
• Overlap management

3. Embedding Stage*

• **Purpose**: Create vector embeddings for semantic search
• **Technology**: OpenAI Embeddings API
• **Features**:
• Batch embedding creation
• Metadata preservation
• Error handling and retry logic
• Performance optimization

4. Storage Stage*

• **Purpose**: Store processed content and embeddings
• **Technology**: PostgreSQL with pgvector extension
• **Features**:
• Vector similarity search
• Metadata storage
• File system integration
• Transaction management

****Concurrency Management****

****Thread Pool Architecture****

• **Executor**: ThreadPoolExecutor with configurable worker count
• **Semaphore**: Controlled concurrency to prevent resource exhaustion
• **Async/Await**: Non-blocking I/O operations
• **Error Isolation**: Individual failures don't affect batch processing

Resource Management*

• **Memory Monitoring**: Track memory usage and adjust concurrency
• **CPU Utilization**: Optimize thread count based on system resources
• **Network Throttling**: Rate limiting for web requests
• **Database Connection Pooling**: Efficient database resource usage

****Error Handling and Resilience****

Retry Logic*

• **Exponential Backoff**: Progressive delay for failed operations
• **Circuit Breaker**: Prevent cascading failures
• **Dead Letter Queue**: Handle permanently failed documents
• **Recovery Mechanisms**: Resume interrupted operations

Monitoring and Logging*

• **Progress Tracking**: Real-time status updates
• **Performance Metrics**: Processing speed and resource usage
• **Error Reporting**: Detailed failure analysis
• **Audit Trail**: Complete operation history

Integration and Deployment

****Environment Requirements****

• **Python**: 3.9 or higher
• **Dependencies**: Docling, PostgreSQL, OpenAI API, Agency Swarm
• **System Resources**: Minimum 8GB RAM, 4 CPU cores
• **Storage**: SSD recommended for optimal performance

****Configuration Management****

• **Environment Variables**: API keys, database connections, processing settings
• **Configuration Files**: Pipeline settings, concurrency parameters
• **Runtime Configuration**: Dynamic adjustment of processing parameters
• **Validation**: Comprehensive configuration validation

****Scalability Considerations****

• **Horizontal Scaling**: Multiple agent instances
• **Load Balancing**: Distribute processing across instances

• **Database Sharding**: Partition data for large-scale operations
• **Caching**: Redis for frequently accessed data

Performance Specifications

Processing Capacity

• **Concurrent Documents**: Up to 50 documents simultaneously
• **Throughput**: 100-500 documents per hour (depending on complexity)
• **Memory Usage**: 2-8GB per concurrent pipeline
• **Storage**: 1-10MB per processed document

Quality Metrics

• **Conversion Accuracy**: 95%+ for standard document formats
• **Image Extraction**: 90%+ success rate
• **Chunking Quality**: Optimized for retrieval performance
• **Embedding Quality**: High-dimensional vector representations

Reliability Targets

• **Uptime**: 99.9% availability
• **Error Rate**: <1% for standard operations
• **Recovery Time**: <5 minutes for service restoration
• **Data Integrity**: 100% data consistency

Future Enhancements

Planned Features

• **Advanced Chunking**: Semantic-aware document segmentation
• **Multi-Modal Processing**: Video and audio document support
• **Real-Time Processing**: Stream processing capabilities
• **Advanced Analytics**: Document processing insights and metrics

Integration Opportunities

• **Cloud Storage**: AWS S3, Google Cloud Storage integration
• **Message Queues**: RabbitMQ, Apache Kafka for async processing
• **Monitoring**: Prometheus, Grafana for operational monitoring
• **API Gateway**: RESTful API for external integrations

Conclusion

The Sage Oracle Agent represents a comprehensive solution for enterprise-scale document processing and knowledge extraction. With its robust architecture, advanced concurrency management, and extensive toolset, it provides organizations with the capability to process large volumes of documents efficiently while maintaining high quality and reliability standards.

The agent's modular design allows for easy customization and extension, making it suitable for a wide range of use cases from research institutions to enterprise knowledge management systems. Its integration with modern technologies like Docling, PostgreSQL, and OpenAI ensures it remains at the forefront of document processing capabilities.

By leveraging the Agency Swarm framework, the Sage Oracle Agent can seamlessly integrate with other specialized agents to create comprehensive document processing workflows that meet the most demanding enterprise requirements.