**Relevant Paragraph Finder - Usage Examples**

**Installation**

1. Install the required packages:

pip install -r requirements.txt

**Command Line Usage**

**Basic usage:**

python relevant\_paragraph\_finder.py document.pdf "machine learning algorithms"

**Save results to file:**

python relevant\_paragraph\_finder.py document.docx "data analysis techniques" -o results.txt

**Different file formats:**

python relevant\_paragraph\_finder.py report.txt "business strategy"  
python relevant\_paragraph\_finder.py thesis.pdf "neural networks"  
python relevant\_paragraph\_finder.py manual.docx "installation guide"

**Python Script Usage**

from relevant\_paragraph\_finder import RelevantParagraphFinder  
  
# Create finder instance  
finder = RelevantParagraphFinder()  
  
# Process a file  
finder.process\_file(  
 file\_path="document.pdf",  
 query="artificial intelligence applications",  
 output\_file="highlighted\_results.txt" # Optional  
)

**How it Works**

1. **File Reading**: Automatically detects file format and extracts text
   * PDF files: Uses PyPDF2 to extract text from all pages
   * DOCX files: Uses python-docx to extract paragraph text
   * Text files: Handles multiple encodings automatically
2. **Paragraph Extraction**: Intelligently splits text into paragraphs
   * Uses double newlines as primary separator
   * Falls back to content-aware splitting for better results
   * Filters out very short paragraphs
3. **Doc2Vec Training**: Creates document embeddings
   * Trains a Doc2Vec model on all paragraphs
   * Uses semantic understanding, not just keyword matching
   * Each paragraph gets a unique vector representation
4. **Similarity Calculation**: Finds most relevant paragraph
   * Converts query to vector using trained model
   * Calculates cosine similarity with all paragraph vectors
   * Returns paragraph with highest similarity score
5. **Highlighting**: Displays results with visual emphasis
   * Console output with colored highlighting (if colorama available)
   * Option to save results to text file
   * Shows similarity scores for transparency

**Supported File Formats**

* ✅ PDF (.pdf) - Requires PyPDF2
* ✅ Word Documents (.docx, .doc) - Requires python-docx
* ✅ Plain Text (.txt)
* ✅ Rich Text Format (.rtf)
* ✅ Markdown (.md)
* ✅ Any text-based format (fallback to plain text reading)

**Example Output**

Processing file: research\_paper.pdf  
Query: machine learning optimization  
--------------------------------------------------  
Extracted 15 paragraphs  
Training Doc2Vec model...  
Doc2Vec model trained on 15 paragraphs  
Finding most relevant paragraph...  
  
================================================================================  
QUERY: machine learning optimization  
MOST RELEVANT PARAGRAPH (Similarity: 0.8742):  
================================================================================  
  
[Paragraph 7]  
Machine learning optimization techniques have revolutionized the field of artificial intelligence by providing robust methods for improving model performance. These techniques include gradient descent variants, adaptive learning rates, and regularization methods that prevent overfitting while maintaining high accuracy on unseen data.  
  
>>> [MOST RELEVANT - Paragraph 7] <<<  
>>> The optimization landscape in machine learning has evolved significantly with the introduction of advanced algorithms such as Adam, RMSprop, and AdaGrad. These optimizers adapt the learning rate dynamically and have proven essential for training deep neural networks effectively. <<<  
  
[Paragraph 8]  
Furthermore, recent developments in optimization have focused on addressing the challenges of non-convex loss surfaces and saddle points that are common in deep learning scenarios.

**Advanced Configuration**

You can modify the Doc2Vec parameters in the code:

# In Doc2VecSimilarity class \_\_init\_\_ method  
def \_\_init\_\_(self, vector\_size=100, window=5, min\_count=1, epochs=20):  
 # vector\_size: Size of the feature vectors  
 # window: Context window size   
 # min\_count: Minimum word frequency  
 # epochs: Number of training iterations

**Troubleshooting**

1. **PDF reading issues**: Make sure PyPDF2 is installed: pip install PyPDF2
2. **DOCX reading issues**: Make sure python-docx is installed: pip install python-docx
3. **No colored output**: Install colorama: pip install colorama
4. **Empty results**: Check if file contains readable text and has sufficient content
5. **Low similarity scores**: Try rephrasing your query or check if document contains relevant content

**Tips for Better Results**

1. **Use specific queries**: More specific queries tend to give better results
2. **Check file quality**: Ensure PDFs have selectable text (not scanned images)
3. **Adjust parameters**: Modify Doc2Vec parameters for different document types
4. **Longer documents work better**: Doc2Vec performs better with more training data