**📄 AI Document Summarizer**

A powerful, intelligent document summarization tool that automatically selects the best AI model based on your document length. Built with state-of-the-art Hugging Face transformers including **LED** for long documents, **BART** for general text, and **T5** for quick processing.

**✨ Features**

* 🤖 **Smart Model Selection**: Automatically chooses the optimal model based on document length
* 📚 **Long Document Support**: Handles documents up to 16,000+ tokens with LED (Longformer Encoder-Decoder)
* ⚡ **Multiple Models**: BART, LED, T5, Pegasus, and Flan-T5 support
* 🔄 **Batch Processing**: Summarize multiple documents efficiently
* 📊 **Advanced Analytics**: Word count, compression ratio, and keyword extraction
* 🚀 **GPU Acceleration**: Automatic GPU detection and usage
* 🛠️ **Easy Integration**: Simple API for quick integration into your projects
* 📈 **Performance Modes**: Choose between speed and quality

**🚀 Quick Start**

**Installation**

# Clone the repository

git clone https://github.com/yourusername/ai-document-summarizer.git

cd ai-document-summarizer

# Install required packages

pip install transformers torch torchvision torchaudio

pip install accelerate sentencepiece

**Basic Usage**

from document\_summarizer import DocumentSummarizer

# Initialize with LED for long documents (recommended)

summarizer = DocumentSummarizer(model\_name="led")

# Summarize your text

text = "Your long document text here..."

summary = summarizer.summarize(text, max\_length=200, min\_length=50)

print(f"Summary: {summary}")

**📖 Documentation**

**Available Models**

| **Model** | **Max Tokens** | **Speed** | **Quality** | **Best For** |
| --- | --- | --- | --- | --- |
| **LED** | 16,384 | Slower | ⭐⭐⭐⭐⭐ | Long documents, research papers |
| **BART** | 1,024 | Fast | ⭐⭐⭐⭐ | News articles, general text |
| **T5** | 512 | Fastest | ⭐⭐⭐ | Quick processing, testing |
| **Pegasus** | 1,024 | Fast | ⭐⭐⭐⭐ | News summarization |
| **Flan-T5** | 512 | Fast | ⭐⭐⭐⭐ | Instruction-following tasks |

**Model Selection Guide**

# For different document types:

summarizer\_short = DocumentSummarizer("t5") # < 300 words

summarizer\_medium = DocumentSummarizer("bart") # 300-1000 words

summarizer\_long = DocumentSummarizer("led") # > 1000 words

**💡 Examples**

**1. Basic Summarization**

from document\_summarizer import DocumentSummarizer

# Initialize summarizer

summarizer = DocumentSummarizer(model\_name="led")

# Your document

document = """

Artificial intelligence (AI) is rapidly transforming industries across the globe...

[Your long document text here]

"""

# Generate summary

summary = summarizer.summarize(

document,

max\_length=150,

min\_length=50

)

print(f"Original length: {len(document.split())} words")

print(f"Summary length: {len(summary.split())} words")

print(f"Summary: {summary}")

**2. Long Document Processing**

# For very long documents (automatic chunking)

long\_summary = summarizer.summarize\_long\_document(

very\_long\_document,

max\_length=300,

min\_length=100

)

**3. Batch Processing**

# Process multiple documents

documents = ["Document 1...", "Document 2...", "Document 3..."]

summaries = summarizer.batch\_summarize(

documents,

max\_length=100,

min\_length=30

)

for i, summary in enumerate(summaries):

print(f"Document {i+1}: {summary}")

**4. Advanced Document Analysis**

from document\_summarizer import AdvancedSummarizer

# Get comprehensive analysis

advanced = AdvancedSummarizer(model\_name="led")

analysis = advanced.analyze\_document(document)

print(f"Summary: {analysis['summary']}")

print(f"Keywords: {analysis['keywords']}")

print(f"Reading time: {analysis['estimated\_reading\_time']}")

print(f"Word count: {analysis['word\_count']}")

**5. Custom Document Processor**

# Process files or direct text

result = process\_your\_document(

"path/to/your/document.txt", # or direct text

model\_name="led",

summary\_length="medium" # short, medium, or long

)

print(f"Source: {result['source']}")

print(f"Compression: {result['compression\_ratio']}")

print(f"Summary: {result['summary']}")

**🎯 Use Cases**

**Academic Research**

# Perfect for research papers

summarizer = DocumentSummarizer("led")

paper\_summary = summarizer.summarize(research\_paper, max\_length=400)

**News Articles**

# Optimized for news content

summarizer = DocumentSummarizer("pegasus")

news\_summary = summarizer.summarize(news\_article, max\_length=100)

**Business Reports**

# Handle long business documents

summarizer = DocumentSummarizer("led")

report\_summary = summarizer.summarize\_long\_document(business\_report)

**Quick Content Processing**

# Fast processing for multiple short texts

summarizer = DocumentSummarizer("t5")

quick\_summaries = summarizer.batch\_summarize(short\_articles)

**⚙️ Configuration**

**Performance Optimization**

# Enable GPU acceleration (automatic detection)

summarizer = DocumentSummarizer("led", device="auto")

# Force CPU usage

summarizer = DocumentSummarizer("led", device="cpu")

# Clear model cache to save memory

summarizer.clear\_cache()

**Custom Length Settings**

# Adjust summary lengths

summary = summarizer.summarize(

text,

max\_length=300, # Maximum summary length

min\_length=100, # Minimum summary length

do\_sample=False # Deterministic vs creative output

)

**📊 Performance Benchmarks**

**Processing Speed (approximate)**

* **T5**: ~500 words/second
* **BART**: ~300 words/second
* **LED**: ~100 words/second
* **Pegasus**: ~250 words/second

**Memory Usage**

* **T5**: ~1GB VRAM
* **BART**: ~2GB VRAM
* **LED**: ~4GB VRAM (handles much longer texts)
* **Pegasus**: ~2GB VRAM

**Quality Metrics (subjective)**

* **LED**: Best for long documents (research papers, reports)
* **BART**: Best overall balance (news, articles, blogs)
* **Pegasus**: Best for news content specifically
* **T5**: Good for quick processing and testing

**🛠️ Requirements**

**System Requirements**

* Python 3.8 or higher
* 4GB+ RAM (8GB+ recommended for LED)
* GPU recommended (CUDA-compatible) for better performance

**Python Dependencies**

torch>=2.0.0

transformers>=4.20.0

accelerate>=0.20.0

sentencepiece>=0.1.99

numpy>=1.21.0

regex>=2022.7.9

**🔧 Installation Options**

**Option 1: Standard Installation**

pip install transformers torch accelerate sentencepiece

**Option 2: With GPU Support**

# For CUDA 11.8

pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu118

pip install transformers accelerate sentencepiece

**Option 3: Development Installation**

git clone https://github.com/yourusername/ai-document-summarizer.git

cd ai-document-summarizer

pip install -r requirements.txt

pip install -e .

**🚨 Troubleshooting**

**Common Issues**

**Model Download Issues:**

# Models are automatically downloaded on first use

# First run may take 5-10 minutes depending on internet speed

# Models are cached locally after first download

**Memory Issues:**

# If you get CUDA out of memory errors:

summarizer = DocumentSummarizer("t5", device="cpu") # Use CPU

# Or use a smaller model like T5 instead of LED

**Slow Performance:**

# Enable GPU if available

summarizer = DocumentSummarizer("led", device="auto")

# Or use faster models for quick processing

summarizer = DocumentSummarizer("t5") # Fastest option

**📚 API Reference**

**DocumentSummarizer Class**

**\_\_init\_\_(model\_name="led", device="auto")**

Initialize the summarizer with specified model.

**Parameters:**

* model\_name (str): Model to use ("led", "bart", "t5", "pegasus", "flan\_t5")
* device (str): Device to use ("auto", "cpu", "cuda")

**summarize(text, max\_length=150, min\_length=50, do\_sample=False)**

Summarize a single document.

**Parameters:**

* text (str): Input text to summarize
* max\_length (int): Maximum length of summary
* min\_length (int): Minimum length of summary
* do\_sample (bool): Whether to use sampling

**Returns:**

* str: Generated summary

**summarize\_long\_document(text, chunk\_size=None, max\_length=150, min\_length=50)**

Summarize long documents with automatic chunking.

**batch\_summarize(texts, max\_length=150, min\_length=50)**

Summarize multiple documents efficiently.

**AdvancedSummarizer Class**

**analyze\_document(text)**

Comprehensive document analysis including summary, keywords, and statistics.

**Returns:**

* dict: Analysis results with summary, keywords, word count, etc.

**🤝 Contributing**

We welcome contributions! Please see our [Contributing Guidelines](https://claude.ai/chat/CONTRIBUTING.md) for details.

**Development Setup**

git clone https://github.com/yourusername/ai-document-summarizer.git

cd ai-document-summarizer

pip install -e ".[dev]"

**Running Tests**

python -m pytest tests/

**📄 License**

This project is licensed under the MIT License - see the [LICENSE](https://claude.ai/chat/LICENSE) file for details.

**🙏 Acknowledgments**

* [Hugging Face](https://huggingface.co/) for the amazing transformers library
* [Facebook Research](https://github.com/facebookresearch) for BART
* [Google Research](https://github.com/google-research) for T5 and Pegasus
* [Allen Institute for AI](https://allenai.org/) for LED (Longformer)

**📞 Support**

* 📧 **Email**: shivamphugat2@gmail.com

**🔮 Roadmap**

* [ ] Add support for more languages
* [ ] Implement abstractive + extractive hybrid summarization
* [ ] Add web interface with Streamlit/Gradio
* [ ] Docker containerization
* [ ] API server with FastAPI
* [ ] Integration with popular document formats (PDF, DOCX)
* [ ] Fine-tuning capabilities for domain-specific summaries

**⭐ Star History**

If this project helps you, please consider giving it a star! ⭐