Evaluating Top LLMs & Summarization Model Recommendations

Provider	Model
OpenAI	"`gpt-3.5-turbo` does pretty well at this task."
Anthropic	"**Claude 3.5 Sonnet** is an excellent choice for use cases such as this where high accuracy is required. If the size and quantity of your documents is large such that costs start to become a concern, you can also try using a smaller model like **Claude 3 Haiku**."
Ollama	"As of September 2023, the 180 billion parameter model, **Falcon 180B**, is the best-performing openly released LLM. It sits somewhere in between OpenAI's GPT 3.5 and GPT 4. For running **Falcon 180B**, a powerful system is recommended with at least 192GB of total memory."
Meta	"**Meta-Llama-3.1-8B-Instruct-Summarizer** : This is a fine-tuned version of Llama 3.1 trained in English, Spanish, and Chinese for text summarization"

Provider	Model
Mistral	“**Mistral 7B with 16k** context for summarization: Version of the Mistral 7B model that has undergone unsupervised fine-tuning for contexts up to 16k.”
Google	“**Gemini Advanced:Understand** large books and reports with of file uploads”
DeepSeek	“**DeepSeek-R1-Distill-Qwen-32B**: " we can safely assume that the DeepSeek-R1-Distill-Qwen-32B is the go-to model for text classification and summarization tasks.".

https://www.summarizer.org/p/chatgpt4o-vs-gemini-advanced

Objective	ChatGPT-4o Output	Gemini Advanced Output	Winner
1: Executive Summary Determine which model better distills key business information for decision-makers.	Detailed summary but needed two revisions for conciseness.	Created a good summary in the first attempt without revisions.	✅ **Gemini Advanced**
2: Academic Summary Evaluate the models' ability to summarize academic content from an essay for students.	Initially too long; the concise version lacked some important details.	Provided a comprehensive summary in the first go without omissions.	✅ **Gemini Advanced**
3: Abstract Generation Assess how well the models generate abstracts for academic papers.	Needed a revision; lacked some contextual information and technical detail.	Generated a comprehensive and detailed abstract on the first try, including key equations.	✅ **Gemini Advanced**
4: Conclusion Generation Evaluate each model's ability to summarize a blog post and present its conclusion.	Provided a concise, clear, and actionable conclusion.	More detailed but less readable and actionable.	✅ **ChatGPT-4o**
5: News Summary See how well the models can summarize news articles for a broad audience.	Detailed but included unnecessary information.	Concise, focused on main events, and clear for a general audience.	✅ **Gemini Advanced**
6: Medical Summary Determine the models' ability to summarize medical information for healthcare providers.	Provided a comprehensive and detailed summary suitable for healthcare professionals.	Concise but lacking in specific details needed by healthcare professionals.	✅ **ChatGPT-4o**

Identifying Access Methods: API Integrations, Local Deployment & More

Provider	API	Local Hosting	Notes
OpenAI	✅	❌	Cloud only (API or UI)
Anthropic	✅	❌	Cloud only
Ollama	❌	✅	Local CLI for model hosting
Meta (LLaMA)	❌	✅	Requires manual setup
Mistral	❌	✅	Open weights, local or cloud
Google (Gemini)	✅	❌	Vertex AI or Bard
DeepSeek	❌	✅	Models on HuggingFace / GitHub

Pricing Models & Licensing Overview

Provider	Pricing	Licensing
OpenAI	Usage-based (tokens)	Closed source
Anthropic	Usage-based	Closed source
Ollama	Free (local inference)	Varies by model
Meta (LLaMA)	Free for research/commercial	Open-ish (license req.)
Mistral	Free (open weights)	Apache 2.0
Google	Usage-based (Vertex AI)	Closed source
DeepSeek	Free (open weights on HF)	Apache 2.0

Can These LLMs Understand Markdown?

Model	Markdown Input Support	Notes
OpenAI GPT	✅	Interprets Markdown well, needs minimal prep
Claude 3	✅	Excels at structured, long-form text
Gemini	✅	Can use formatting cues for better summarization
Ollama	⚠️ Varies	Depends on chosen model, best with formatting stripped
Meta LLaMA	⚠️ Prompt-tuned	Preprocessing likely needed
Mistral	⚠️ Limited	Markdown support unclear, best used with plain text
DeepSeek	⚠️ Limited	Works better with cleaned/plain transcripts

Use Case	Recommended Models
Best for Markdown Input	Claude 3, GPT-4
Best Free Local Options	DeepSeek, Mistral, LLaMA2 via Ollama
Simple API Access	OpenAI, Claude, Gemini
Fully Open-Source Stack	DeepSeek, Mistral, Meta (LLaMA)

⭐ A highly starred Medium blog by Isaac Tham titled **"Summarize Podcast Transcripts and Long Texts Better with NLP and AI"**, along with the accompanying GitHub code

Why Chunking Matters in Summarizing Long Texts

Summarized from the reference

- A common way to summarize long text is by **splitting it into smaller chunks**, summarizing each, and combining them (recursive summarization).
- Many tools, like LangChain's `map_reduce`, **split the text without considering logical or structural flow**.
    - **Problem: Important information may be split awkwardly and lose its meaning.**
- For example:
    - A 1000-word article with 200-word chunks creates 5 chunks.
    - If the author’s first main point takes 250 words, the last 50 words get mixed with the next topic.
    - **Problem: Mixing topics in chunks can confuse the summarizer and lead to missing key points.**

This is a summary based on the reference.

- Another method is the *refine* method:
    - It updates the summary step-by-step as it reads new chunks.
    - **Problem: It runs sequentially (not in parallel), making it slower.**
    - **Problem: It tends to focus too much on the beginning of the text**, which can be misleading—especially for things like podcast transcripts where intros or ads aren't important.
- Even if future models can read more at once:
    - **Problem: They still can't handle extremely long texts like books without chunking.**
- Final takeaway:
    - **Problem: If a summarization method doesn't understand the meaning and structure of the text, the final summary won't reflect the author's intent properly.**

Simple Summary: Summarized from the reference

A better way to summarize long texts is to **combine summarization with topic modeling** in the same process. Instead of blindly splitting the text, we **group similar parts (chunks) by topic** and then pass these grouped topics into the next summarization step. This helps keep the meaning clear and organized.

You can do this in Python using:

- `scipy` – to measure how similar text chunks are (cosine distance),
- `networkx` – to group similar chunks using the Louvain algorithm,
- `langchain` – to easily use language models like GPT-3.

This sis a summary based on the reference.

The first step involves splitting the raw text into individual sentences while filtering out sentences that are too short or too long—only those between 20 and 80 words are kept to maintain meaningful content.

Next comes the chunking process. Instead of creating chunks based solely on what fits inside an LLM's context window, the approach here is to chunk the text by ideas. Each chunk is made up of about five sentences, with a one-sentence overlap between chunks to preserve context and flow. This method leads to more coherent chunks, each ideally representing a distinct idea. For the speech used in this example, this resulted in 65 chunks, with an average chunk length of 148 words.

After chunking, the summarization begins. Each chunk is passed through a custom prompt using GPT-3 via LangChain, which returns both a summary and a short, informative title for the chunk. This two-in-one output is especially useful for organizing and clustering chunks by topic in the next steps. The use of descriptive titles helps improve the clarity and structure of the final summary, while also making it easier to group similar content.

Summarized from the reference.

Instead of summarizing blindly, the method uses topic modeling and semantic clustering to organize chunks by meaning before passing them to the next summarization level. This ensures that related ideas stay together, helping to preserve the author's original intent. The clustering is done using tools like `scipy` for measuring similarity, `networkx` for community detection, and `langchain` for LLM calls. This combined approach results in more accurate, structured, and meaningful summaries, even for long texts that wouldn’t fit into a model’s context window all at once.

https://arxiv.org/abs/2305.14239

On Learning to Summarize with Large Language Models as References

Yixin Liu[1] Kejian Shi[1] Katherine S He[1] Longtian Ye[1]
Alexander R. Fabbri[2] Pengfei Liu[3] Dragomir Radev[1] Arman Cohan[1,4]
[1]Yale University [2]Salesforce AI [3]Shanghai Jiao Tong University [4]Allen Institute for AI
{yixin.liu,arman.cohan}@yale.edu

🧠 MLE Fine-Tuning:

The model is trained using example summaries created by a large language model (LLM). It learns to copy those summaries by predicting each word step by step. This method is called **Maximum Likelihood Estimation (MLE)**.

💡 Contrastive Learning:

To make the model even better, another method is used called **contrastive learning**. Instead of just copying one good summary, the model looks at many possible summaries. It learns to give **higher scores to better summaries** and **lower scores to worse ones**. “Better” here means more similar to the reference, based on a quality score like ROUGE.

This helps the model understand what a good summary looks like, not just memorize one version.

🔸 What is ROUGE?

ROUGE stands for **Recall-Oriented Understudy for Gisting Evaluation**. It's a way to **measure how similar a generated summary is to a reference summary** by comparing words and phrases.

🟠 ROUGE-1:

- Looks at **single words** (called **unigrams**).
- Example: If the reference summary has the word "science" and your summary also has "science," that's a match.
- ROUGE-1 measures **how many of those single-word overlaps there are** between the two summaries.

🔵 ROUGE-2:

- Looks at **two-word phrases** (called **bigrams**).
- Example: If the reference says "climate change" and your summary also says "climate change," that's a ROUGE-2 match.
- It checks how many two-word combinations match between summaries.

🟢 **F1 Score (in ROUGE-1/2 F1):**

- Combines **precision** (how much of what you wrote was correct) and **recall** (how much of the reference you covered).
- **F1 score** balances both — so you're rewarded for being accurate **and** complete.

📌 **Summary:**

- **ROUGE-1** = match of single words
- **ROUGE-2** = match of two-word phrases
- **F1 score** = balance of precision and recall

The **higher the ROUGE-1/2 F1 score**, the more your summary looks like the reference.

Large Language Models (LLMs) like GPT can do more than just give good example summaries — they can also **judge the quality** of any summary. These new LLM-based evaluation methods are **more accurate** than older ones like ROUGE.

So, instead of using only example summaries for training, the model can learn by **ranking summaries** based on how good the LLM thinks they are. This helps train the model better and faster.

They use two LLM-based methods for this:

- **GPTScore**: It scores how likely a summary is, according to the LLM.
- **GPTRank**: An improved version of another method called G-Eval.

The model then learns to give **higher scores to better summaries**, based on these LLM evaluations.

System	LP	GS	R1	R2	Len.
GPT3D3	-22.62	-0.271	100.0	100.0	85.4
BART	-59.55	-0.789	46.85	24.38	79.0
GPT3D2	-41.21	-0.547	55.40	33.72	78.7
Alpaca	-44.82	-0.567	51.53	30.18	81.8
GPT-3.5	-45.12	-0.498	58.14	37.46	92.0
BART.GPT3D3	-36.13	-0.420	**59.50**	**40.70**	85.6
BRIO.GPT3D3	**-26.20**	**-0.318**	56.21	36.47	83.7

Table 1: Results with GPTScore. **LP** is the log-probability predicted by **GPT3D3**. **GS** is the GPTScore based on GPT3D3. **R1** and **R2** are the ROUGE1/2 F1 scores respectively. **Len.** is the average summary length. **BART.GPT3D3** is fine-tuned with MLE training while **BRIO.GPT3D3** is fine-tuned with contrastive learning.

1. **Training with MLE using LLM-generated summaries** improves the original BART model's performance, based on GPTScore and ROUGE metrics.

2. **Contrastive learning** leads to even **better performance** (measured by GPTScore) than MLE training. This shows that contrastive learning helps the model better understand and match the patterns of the LLM.

3. The contrastively trained model (**BRIO.GPT3D3**) performs **almost as well as the LLM itself**, even though it was trained on only **100 examples**. This shows that contrastive learning is **very efficient and promising**.

Another summarization evaluation metric from Isaac Tham's blog

In addition to using the ROUGE score, you can try **DeepEval's summarization metric** as a good starting point to evaluate summary quality.

It looks at two things:

- **Alignment**: The LLM checks if the claims in the summary are actually supported by the original text.
- **Coverage**: The LLM creates questions based on the original text and tries to answer them using only the summary. If the summary doesn't have the answer, the LLM says "idk."

The **final score** is the lower of the two: alignment or coverage.

Isaac Tham also provides another evaluation metric in his summarization evaluation blog that I haven't mentioned here. If you're interested, you can check out the full blog for more details.