**Question:** how to improve LLM's ability in generating better quality summary of a document?

Improving the ability of a large language model (LLM) like GPT or others to generate higher-quality summaries of a document involves several strategies that optimize **how the model processes and synthesizes the input.** Here are several approaches to enhance the summary quality:

**### 1. \*\*Provide Clear Instructions in the Prompt\*\***

**The quality of the summary largely depends on the prompt design**. A well-structured prompt with clear instructions can guide the LLM to focus on the key aspects of the document.

#### Example:

Instead of a simple prompt like:

> Summarize this document.

**Use a more detailed prompt that specifies the summary length, important points, and style:**

> Summarize the key points of this document in 3-4 sentences. Focus on the main arguments and the conclusions. Avoid unnecessary details and maintain an objective tone.

**### 2. \*\*Break Long Documents into Smaller Chunks\*\***

LLMs can struggle with large amounts of text in one go, as they have a limit on how much text they can process (token limit). **By breaking the document into logical sections and summarizing each part, you improve the accuracy and relevance of the summaries.**

#### Approach:

- Divide the document into sections or paragraphs.

- Generate summaries for each section, then combine them into a cohesive summary.

- Optionally, ask the model to refine the final summary.

#### Example:

> Summarize the following section of the document:

[Section of the document]

Then combine summaries of all sections into one final summary.

**### 3. \*\*Use a Step-by-Step Approach (Structured Summarization)\*\***

Guide the model to extract information systematically, such as asking it to first identify the main points, then the supporting details, and finally, to generate a summary from this structured information.

#### Example:

1. \*\*Step 1:\*\* Ask the LLM to list the key points of the document.

> List the 5 most important points discussed in this document.

2. \*\*Step 2:\*\* Ask it to summarize the key points into a concise paragraph.

> Summarize the key points into a short paragraph.

This structured process helps prevent the model from omitting critical details or becoming too verbose.

### 4. \*\*Use Retrieval-Augmented Generation (RAG)\*\*

For large or complex documents, you can use a retrieval-augmented approach to ensure that the LLM is referencing the right parts of the document. In Retrieval-Augmented Generation (RAG), the system retrieves relevant sections of the document before asking the model to generate a summary.

#### How It Works:

- **Use a search or retrieval system (such as embeddings and vector similarity search) to retrieve the most relevant sections of the document based on the query.**

- Feed the retrieved sections to the LLM to summarize them.

**### 5. \*\*Incorporate Grounding and Fact-Checking\*\***

LLMs can sometimes hallucinate facts or deviate from the source document. To improve factual accuracy:

- \*\*Grounding\*\*: Make sure the model refers to specific parts of the document. This can be done by breaking the input into chunks or by using references in the document (e.g., section titles).

- \*\*Fact-checking\*\*: After generating the summary, use another round of verification to ensure that the summary accurately reflects the content of the document.

#### Example Prompt:

> Summarize the main arguments in the following section. Make sure the summary only includes facts mentioned in the text:

[Section of document]

### 6. \*\*Use Evaluation and Iterative Refinement\*\*

You can improve summary quality by having the model evaluate or refine its own outputs iteratively. After generating an initial summary, you can ask the model to review or improve it based on certain criteria (e.g., clarity, conciseness, coverage).

#### Example Prompts:

- \*\*Initial Summary\*\*:

> Summarize the document in a concise paragraph.

- \*\*Refinement\*\*:

> Improve the above summary to make it clearer and more concise. Focus only on the key points.

You can also use evaluation metrics (e.g., comprehensiveness, relevance) to guide the refinement process.

**### 7. \*\*Tailor the Summary to a Specific Audience\*\***

The LLM can provide better-quality summaries if you specify the intended audience. For example, you might ask the model to summarize the document for a technical expert versus a general audience, which affects the level of detail and complexity in the summary.

#### Example Prompts:

- \*\*For a technical audience\*\*:

> Summarize the key findings of this research paper in 5 sentences, focusing on technical details for a scientific audience.

- \*\*For a general audience\*\*:

> Summarize this article in simple language for someone with no technical background. Keep the summary under 100 words.

**### 8. \*\*Optimize Input Formatting\*\***

The formatting and structure of the input text can impact the model’s ability to generate an accurate summary. Ensure that the document is well-organized with clearly defined sections, headers, and bullet points. LLMs perform better when the text is structured logically.

#### Example:

- If the document has clear sections such as \*\*Introduction\*\*, \*\*Methods\*\*, \*\*Results\*\*, and \*\*Conclusion\*\*, it's better to provide the model with individual sections or guide it to summarize based on these divisions.

### 9. \*\*Use Post-Summarization Filters\*\*

After generating the summary, you can apply filters or constraints to ensure the output meets certain quality standards. For instance:

- \*\*Length constraints\*\*: Ensure the summary stays within the desired length.

- \*\*Redundancy removal\*\*: Remove repetitive or redundant information from the summary.

- \*\*Content checks\*\*: Use additional prompts to check if critical points are missing.

### 10. \*\*Fine-Tune the Model on Summarization Tasks\*\*

For more advanced users, you can fine-tune a large language model specifically for summarization tasks by training it on domain-specific summarization datasets. This helps the model better understand the structure of summaries in a specific context (e.g., scientific papers, news articles).

### Example Process for Fine-Tuning:

- Collect a large dataset of documents and their corresponding human-written summaries.

- Fine-tune the pre-trained LLM using this dataset.

- Evaluate the fine-tuned model on a test set and adjust the model as needed.

This approach is especially useful when the domain or style of the summary is important, such as medical or legal documents.

### 11. \*\*Use Metrics to Evaluate Summaries\*\*

Automated metrics like ROUGE, BLEU, or human evaluations can be used to gauge the quality of the generated summaries. By measuring factors like \*\*completeness\*\*, \*\*clarity\*\*, and \*\*conciseness\*\*, you can track improvements in summary generation over time.

### Conclusion

Improving the quality of LLM-generated summaries involves a combination of prompt engineering, retrieval techniques, breaking the document into smaller parts, iterative refinement, and, in some cases, fine-tuning. By applying these strategies, you can guide the model to produce summaries that are more accurate, concise, and relevant to the document’s content.

Here’s a prompt designed for Anthropic’s Claude to generate a summary of a Federal Reserve official’s speech, along with key metrics to measure its quality and completeness:

---

\*\*Prompt:\*\*

I want you to generate a concise summary of a recent Federal Reserve official’s speech. The summary should capture the main points, policy implications, and economic outlook discussed by the official.

Please also include the following metrics to evaluate the generated summary:

1. \*\*Comprehensiveness:\*\* Does the summary cover all key themes (e.g., interest rate changes, inflation targets, labor market conditions, etc.)? Rate on a scale of 1 to 5.

2. \*\*Clarity:\*\* Is the summary easy to understand and free from unnecessary jargon? Rate on a scale of 1 to 5.

3. \*\*Accuracy:\*\* How accurately does the summary reflect the official’s speech? Rate on a scale of 1 to 5.

4. \*\*Neutrality:\*\* Is the summary unbiased and objective in its tone? Rate on a scale of 1 to 5.

5. \*\*Readability:\*\* Is the summary well-structured and easy to read? Rate on a scale of 1 to 5.

First, generate the summary of the speech, then provide the metrics evaluation.

---

This prompt asks Claude to not only summarize but also assess the summary’s quality on multiple dimensions, providing a holistic output.

To measure the above metrics—\*\*Comprehensiveness\*\*, \*\*Clarity\*\*, \*\*Accuracy\*\*, \*\*Neutrality\*\*, and \*\*Readability\*\*—we can develop a semi-automated approach. Here's a breakdown of how each metric can be measured and a corresponding Python-based implementation using Natural Language Processing (NLP) techniques.

### Approach:

1. \*\*Comprehensiveness\*\*:

- Use \*\*keyword matching\*\* or \*\*topic modeling\*\* (e.g., Latent Dirichlet Allocation - LDA) to identify if the key themes from the official’s speech are covered in the summary.

- Compare the number of relevant topics/keywords found in both the speech and summary.

2. \*\*Clarity\*\*:

- Measure using readability metrics like \*\*Flesch-Kincaid\*\* or \*\*Gunning Fog Index\*\*, which assess sentence length and complexity.

3. \*\*Accuracy\*\*:

- This can be harder to automate, but using \*\*semantic similarity\*\* (cosine similarity between embeddings) between the speech and summary can provide a proxy for accuracy. We can use \*\*BERT\*\* or other sentence transformers for this.

4. \*\*Neutrality\*\*:

- Sentiment analysis tools (e.g., \*\*VADER\*\*, \*\*TextBlob\*\*) can be used to detect any bias or strong emotional language in the summary.

5. \*\*Readability\*\*:

- Readability metrics, again like \*\*Flesch-Kincaid\*\*, can be used to measure how easy the text is to read.

---

### Python Code Implementation

Below is the Python code that uses `spaCy`, `transformers`, `textstat`, and `nltk` to measure these metrics:

```python

import spacy

from sklearn.metrics.pairwise import cosine\_similarity

from sentence\_transformers import SentenceTransformer

import textstat

from nltk.sentiment import SentimentIntensityAnalyzer

# Load language model

nlp = spacy.load("en\_core\_web\_sm")

# Load BERT model for embeddings

model = SentenceTransformer('all-MiniLM-L6-v2')

# Sentiment Analyzer

sia = SentimentIntensityAnalyzer()

# Speech and summary inputs

speech = "The Federal Reserve will continue to monitor inflation closely..."

summary = "The Fed is focusing on inflation and will take action if needed..."

# 1. Comprehensiveness (Keyword/Topic Coverage)

def measure\_comprehensiveness(speech, summary):

doc\_speech = nlp(speech)

doc\_summary = nlp(summary)

# Extract key nouns and topics

keywords\_speech = {chunk.text for chunk in doc\_speech.noun\_chunks}

keywords\_summary = {chunk.text for chunk in doc\_summary.noun\_chunks}

# Check coverage of speech keywords in the summary

coverage = len(keywords\_summary.intersection(keywords\_speech)) / len(keywords\_speech)

return round(coverage \* 5, 2) # Scale 1 to 5

# 2. Clarity (Readability Score)

def measure\_clarity(summary):

flesch\_reading\_ease = textstat.flesch\_reading\_ease(summary)

# Convert Flesch score to 1-5 scale (higher score = easier readability)

if flesch\_reading\_ease >= 60:

return 5

elif flesch\_reading\_ease >= 50:

return 4

elif flesch\_reading\_ease >= 40:

return 3

elif flesch\_reading\_ease >= 30:

return 2

else:

return 1

# 3. Accuracy (Cosine Similarity)

def measure\_accuracy(speech, summary):

speech\_embedding = model.encode([speech])

summary\_embedding = model.encode([summary])

similarity = cosine\_similarity(speech\_embedding, summary\_embedding)[0][0]

return round(similarity \* 5, 2) # Scale 1 to 5

# 4. Neutrality (Sentiment Analysis)

def measure\_neutrality(summary):

sentiment = sia.polarity\_scores(summary)

# If compound sentiment is close to 0, it's neutral

neutrality\_score = 5 - abs(sentiment['compound'] \* 5)

return round(neutrality\_score, 2)

# 5. Readability (Readability Metric)

def measure\_readability(summary):

flesch\_kincaid\_grade = textstat.flesch\_kincaid\_grade(summary)

# Convert to 1-5 scale (lower grade level = more readable)

if flesch\_kincaid\_grade <= 5:

return 5

elif flesch\_kincaid\_grade <= 8:

return 4

elif flesch\_kincaid\_grade <= 10:

return 3

elif flesch\_kincaid\_grade <= 12:

return 2

else:

return 1

# Calculate all metrics

metrics = {

"Comprehensiveness": measure\_comprehensiveness(speech, summary),

"Clarity": measure\_clarity(summary),

"Accuracy": measure\_accuracy(speech, summary),

"Neutrality": measure\_neutrality(summary),

"Readability": measure\_readability(summary),

}

# Display the results

import ace\_tools as tools; tools.display\_dataframe\_to\_user(name="Summary Quality Metrics", dataframe=pd.DataFrame([metrics]))

```

---

### How Each Metric is Measured:

1. \*\*Comprehensiveness\*\*: By checking the overlap of keywords and noun phrases between the speech and summary.

2. \*\*Clarity\*\*: Using the \*\*Flesch-Kincaid Reading Ease\*\* metric, which measures the complexity of the text.

3. \*\*Accuracy\*\*: Using \*\*cosine similarity\*\* between the embeddings of the speech and summary (BERT embeddings).

4. \*\*Neutrality\*\*: Using \*\*sentiment analysis\*\* to assess the bias or emotional tone of the summary.

5. \*\*Readability\*\*: Using the \*\*Flesch-Kincaid Grade Level\*\* to evaluate how readable the summary is.

---

This approach provides a semi-automated method to measure the quality of the generated summary based on the specified metrics. You can refine these methods further depending on the depth of analysis needed.