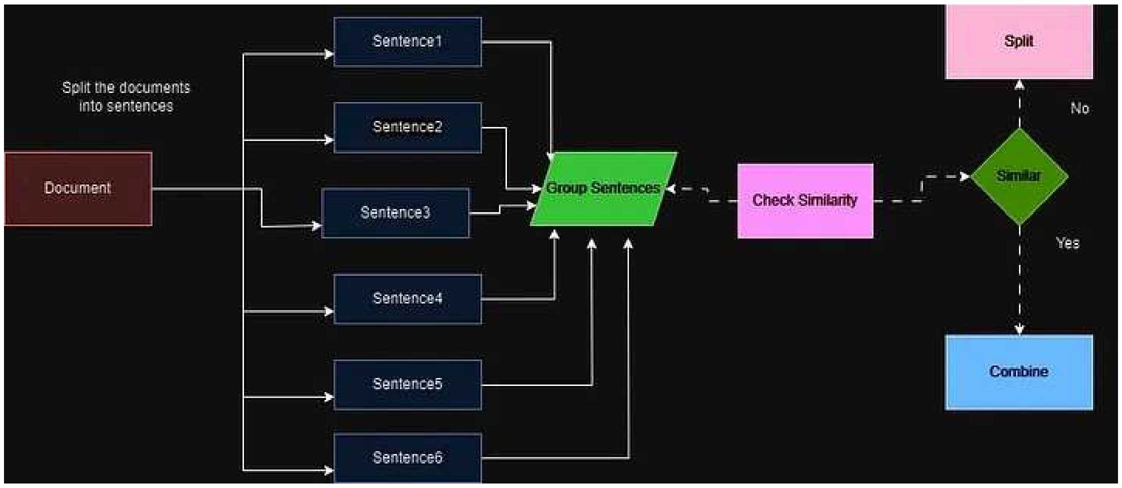
Semantic Chunking for RAG

Plaban Nayak · Following
Published in The AI Forum · 17 min read · Apr 21, 2024



What is Chunking ?

In order to abide by the context window of the LLM , we usually break text into smaller parts / pieces which is called chunking.

What is RAG?

LLMs, although capable of generating text that is both meaningful and grammatically correct, these LLMs suffer from a problem called hallucination. Hallucination in LLMs is the concept where the LLMs confidently generate wrong answers, that is they make up wrong answers in a way that makes us believe that it is true. This has been a major problem since the introduction of the LLMs. These hallucinations lead to incorrect and factually wrong answers. Hence Retrieval Augmented Generation was introduced.

In RAG, we take a list of documents/chunks of documents and encode these textual documents into a numerical representation called vector embeddings, where a single vector embedding represents a single chunk of document and stores them in a database called vector store. The models required for encoding these chunks into embeddings are called encoding models or bi-encoders. These encoders are trained on a large corpus of data, thus making them powerful enough to encode the chunks of documents in a single vector embedding representation.

The retrieval greatly depends on how the chunks are manifested and stored in the vectorstore. Finding the right chunk size for any given text is a very hard question in general.

Improving Retrieval can be done by various retrieval method. But it can also be done by better chunking strategy.

Different chunking methods:

- Fixed size chunking
- Recursive Chunking
- Document Specific Chunking
- Semantic Chunking
- Agentic Chunking

Fixed Size Chunking: This is the most common and straightforward approach to chunking: we simply decide the number of tokens in our chunk and, optionally, whether there should be any overlap between them. In general, we will want to keep some overlap between chunks to make sure that the semantic context doesn't get lost between chunks. Fixed-sized chunking will be the best path in most common cases. Compared to other forms of chunking, fixed-sized chunking is computationally cheap and simple to use since it doesn't require the use of any NLP libraries.

Recursive Chunking : Recursive chunking divides the input text into smaller chunks in a hierarchical and iterative manner using a set of separators. If the initial attempt at splitting the text doesn’t produce chunks of the desired size or structure, the method recursively calls itself on the resulting chunks with a different separator or criterion until the desired chunk size or structure is achieved. This means that while the chunks aren’t going to be exactly the same size, they’ll still “aspire” to be of a similar size. Leverages what is good about fixed size chunk and overlap.

Document Specific Chunking: It takes into consideration the structure of the document . Instead of using a set number of characters or recursive process it creates chunks that align with the logical sections of the document like paragraphs or sub sections. By doing this it maintains the author’s organization of the content thereby keeping the text coherent. It makes the retrieved information more relevant and useful, particularly for structured documents with clearly defined sections. It can handle formats such as Markdown, Html, etc.

Sematic Chunking: Semantic Chunking considers the relationships within the text. It divides the text into meaningful, semantically complete chunks. This approach ensures the information’s integrity during retrieval, leading to a more accurate and contextually appropriate outcome. It is slower compared to the previous chunking strategy

Agentic Chunk: The hypothesis here is to process documents in a fashion that humans would do.

1. We start at the top of the document, treating the first part as a chunk.
2. We continue down the document, deciding if a new sentence or piece of information belongs with the first chunk or should start a new one
3. We keep this up until we reach the end of the document.

This approach is still being tested and isn't quite ready for the big leagues due to the time it takes to process multiple LLM calls and the cost of those calls. There's no implementation available in public libraries just yet.

Here we will experiment with Semantic chunking and Recursive Retriever .

Comparison of methods steps:

1. Load the Document
2. Chunk the Document using the following two methods: Semantic chunking and Recursive Retriever .
3. Assess qualitative and quantitative improvements with RAGAS

Semantic Chunks

Semantic chunking involves taking the embeddings of every sentence in the document, comparing the similarity of all sentences with each other, and then grouping sentences with the most similar embeddings together.

By focusing on the text's meaning and context, Semantic Chunking significantly enhances the quality of retrieval. It's a top-notch choice when maintaining the semantic integrity of the text is vital.

The hypothesis here is we can use embeddings of individual sentences to make more meaningful chunks. Basic idea is as follows :-

1. *Split the documents into sentences based on separators(.,?,!)*
2. *Index each sentence based on position.*
3. *Group: Choose how many sentences to be on either side. Add a buffer of sentences on either side of our selected sentence.*
4. *Calculate distance between group of sentences.*
5. *Merge groups based on similarity i.e. keep similar sentences together.*
6. *Split the sentences that are not similar.*

Technology Stack Used

- **Langchain** :LangChain is an open-source framework designed to simplify the creation of applications using large language models (LLMs). It provides a standard interface for chains, lots of integrations with other tools, and end-to-end chains for common applications.

- **LLM**: Groq's **Language Processing Unit (LPU)** is a cutting-edge technology designed to significantly enhance AI computing performance, especially for Large Language Models (LLMs). The primary goal of the Groq LPU system is to provide real-time, low-latency experiences with exceptional inference performance.
- **Embedding Model**: FastEmbed is a lightweight, fast, Python library built for embedding generation.
- **Evaluation:** Ragas offers metrics tailored for evaluating each component of your RAG pipeline in isolation.

Code Implementation

Install the required dependencies

```
ngchain_community langchain ragas chromadb langchain-groq fastembed pypdf openai
```

```
langchain==0.1.16

langchain-community==0.0.34

langchain-core==0.1.45

langchain-experimental==0.0.57

langchain-groq==0.1.2

langchain-openai==0.1.3

langchain-text-splitters==0.0.1

langcodes==3.3.0

langsmith==0.1.49

chromadb==0.4.24

ragas==0.1.7

fastembed==0.2.6
```

Download data

```
! wget "https://arxiv.org/pdf/1810.04805.pdf"
```

Process the PDF Content

```
from langchain.document_loaders import PyPDFLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter
#
loader = PyPDFLoader("1810.04805.pdf")
documents = loader.load()
#
print(len(documents))
```

Perform Native Chunking(RecursiveCharacterTextSplitting)

```
from langchain.text_splitter import RecursiveCharacterTextSplitter

text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=1000,
    chunk_overlap=0,
    length_function=len,
    is_separator_regex=False
)
#
naive_chunks = text_splitter.split_documents(documents)
for chunk in naive_chunks[10:15]:
  print(chunk.page_content+ "\n")

###########################RESPONSE###############################
BERT BERT
E[CLS] E1 E[SEP] ... ENE1'... EM'
C
T1
T[SEP] ...
 TN
T1'...
 TM'
[CLS] Tok 1 [SEP] ... Tok NTok 1 ... TokM
Question Paragraph Start/End Span
BERT
E[CLS] E1 E[SEP] ... ENE1'... EM'
C
T1
T[SEP] ...
 TN
T1'...
 TM'
[CLS] Tok 1 [SEP] ... Tok NTok 1 ... TokM
```

Masked Sentence A Masked Sentence B
Pre-training Fine-Tuning NSP Mask LM Mask LM
Unlabeled Sentence A and B Pair SQuAD
Question Answer Pair NER MNLI Figure 1: Overall pre-training and ﬁne-tuning proc
tures are used in both pre-training and ﬁne-tuning. The same pre-trained model p
models for different down-stream tasks. During ﬁne-tuning, all parameters are ﬁn
symbol added in front of every input example, and [SEP] is a special separator t
tions/answers).
ing and auto-encoder objectives have been used
for pre-training such models (Howard and Ruder,

2018; Radford et al., 2018; Dai and Le, 2015).
2.3 Transfer Learning from Supervised Data
There has also been work showing effective trans-
fer from supervised tasks with large datasets, such
as natural language inference (Conneau et al.,
2017) and machine translation (McCann et al.,
2017). Computer vision research has also demon-
strated the importance of transfer learning from
large pre-trained models, where an effective recipe
is to ﬁne-tune models pre-trained with Ima-
geNet (Deng et al., 2009; Yosinski et al., 2014).
3 BERT
We introduce BERT and its detailed implementa-
tion in this section. There are two steps in our
framework: pre-training and ﬁne-tuning . Dur-
ing pre-training, the model is trained on unlabeled
data over different pre-training tasks. For ﬁne-
tuning, the BERT model is ﬁrst initialized with
the pre-trained parameters, and all of the param-
eters are ﬁne-tuned using labeled data from the
downstream tasks. Each downstream task has sep-

arate ﬁne-tuned models, even though they are ini-
tialized with the same pre-trained parameters. The
question-answering example in Figure 1 will serve
as a running example for this section.
A distinctive feature of BERT is its uniﬁed ar-
chitecture across different tasks. There is mini-mal difference between the pre-
ture and the ﬁnal downstream architecture.
Model Architecture BERT’s model architec-
ture is a multi-layer bidirectional Transformer en-
coder based on the original implementation de-
scribed in Vaswani et al. (2017) and released in
thetensor2tensor library.1Because the use
of Transformers has become common and our im-
plementation is almost identical to the original,
we will omit an exhaustive background descrip-
tion of the model architecture and refer readers to
Vaswani et al. (2017) as well as excellent guides
such as “The Annotated Transformer.”2
In this work, we denote the number of layers
(i.e., Transformer blocks) as L, the hidden size as

H, and the number of self-attention heads as A.3
We primarily report results on two model sizes:
BERT BASE (L=12, H=768, A=12, Total Param-
eters=110M) and BERT LARGE (L=24, H=1024,
A=16, Total Parameters=340M).
BERT BASE was chosen to have the same model
size as OpenAI GPT for comparison purposes.

```
Critically, however, the BERT Transformer uses
bidirectional self-attention, while the GPT Trans-
former uses constrained self-attention where every
token can only attend to context to its left.4
1https://github.com/tensorﬂow/tensor2tensor
2http://nlp.seas.harvard.edu/2018/04/03/attention.html
3In all cases we set the feed-forward/ﬁlter size to be 4H,
i.e., 3072 for the H= 768 and 4096 for the H= 1024 .
4We note that in the literature the bidirectional Trans-

Input/Output Representations To make BERT
handle a variety of down-stream tasks, our input
representation is able to unambiguously represent
both a single sentence and a pair of sentences
(e.g.,⟨Question, Answer⟩) in one token sequence.
Throughout this work, a “sentence” can be an arbi-
trary span of contiguous text, rather than an actual
linguistic sentence. A “sequence” refers to the in-
put token sequence to BERT, which may be a sin-
gle sentence or two sentences packed together.
We use WordPiece embeddings (Wu et al.,
2016) with a 30,000 token vocabulary. The ﬁrst
token of every sequence is always a special clas-
siﬁcation token ( [CLS] ). The ﬁnal hidden state
corresponding to this token is used as the ag-
gregate sequence representation for classiﬁcation
tasks. Sentence pairs are packed together into a
single sequence. We differentiate the sentences in
two ways. First, we separate them with a special
token ( [SEP] ). Second, we add a learned embed-
```

Instantiate Embedding Model

```
from langchain_community.embeddings.fastembed import FastEmbedEmbeddings
embed_model = FastEmbedEmbeddings(model_name="BAAI/bge-base-en-v1.5")
```

Setup the API Key for LLM

```
from google.colab import userdata
from groq import Groq
from langchain_groq import ChatGroq
#
groq_api_key = userdata.get("GROQ_API_KEY")
```

Perform Semantic Chunking

We're going to be using the `percentile` threshold as an example today — but there's three different strategies you could use on Semantic Chunking):

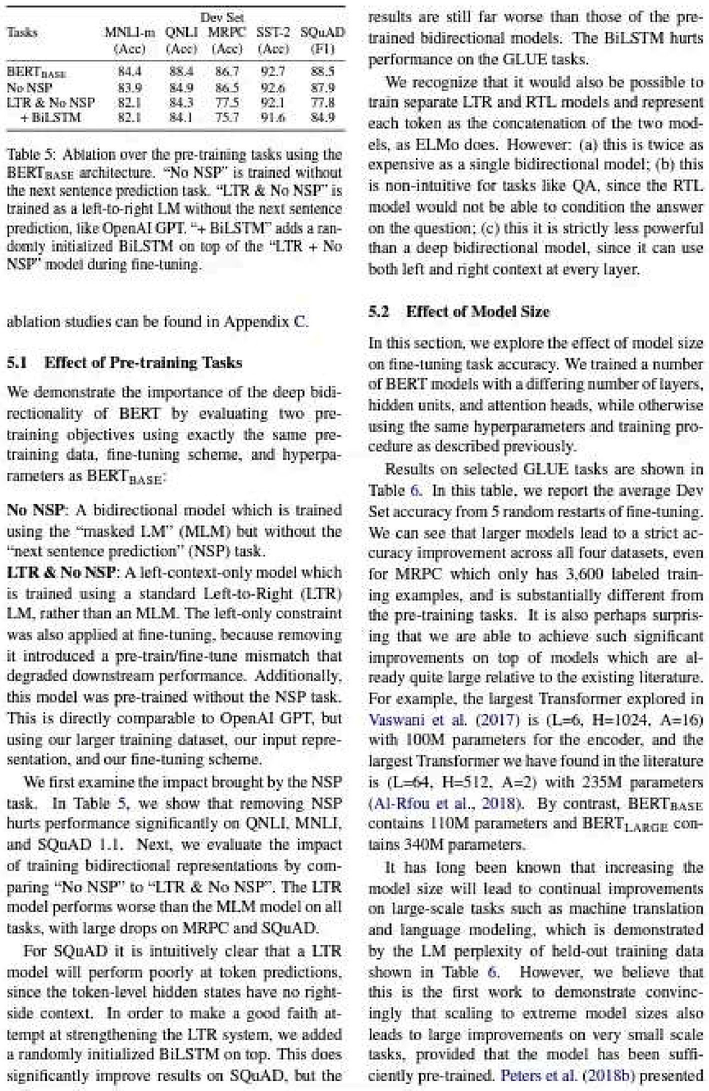
- `**percentile**` (default) — In this method, all differences between sentences are calculated, and then any difference greater than the X percentile is split.

*- `**standard_deviation**` **—** In this method, any difference greater than X standard deviations is split.*

- `**interquartile**` — In this method, the interquartile distance is used to split chunks.

NOTE: This method is currently experimental and is not in a stable final form — expect updates and improvements in the coming months

```
from langchain_experimental.text_splitter import SemanticChunker
from langchain_openai.embeddings import OpenAIEmbeddings

semantic_chunker = SemanticChunker(embed_model, breakpoint_threshold_type="perce
#
semantic_chunks = semantic_chunker.create_documents([d.page_content for d in doc
#
for semantic_chunk in semantic_chunks:
  if "Effect of Pre-training Tasks" in semantic_chunk.page_content:
    print(semantic_chunk.page_content)
    print(len(semantic_chunk.page_content))

#############################RESPONSE##############################
Dev Set
Tasks MNLI-m QNLI MRPC SST-2 SQuAD
(Acc) (Acc) (Acc) (Acc) (F1)
BERT BASE 84.4 88.4 86.7 92.7 88.5
No NSP 83.9 84.9 86.5 92.6 87.9
LTR & No NSP 82.1 84.3 77.5 92.1 77.8
+ BiLSTM 82.1 84.1 75.7 91.6 84.9
Table 5: Ablation over the pre-training tasks using the
BERT BASE architecture. “No NSP” is trained without
the next sentence prediction task. “LTR & No NSP” is
trained as a left-to-right LM without the next sentence
prediction, like OpenAI GPT. “+ BiLSTM” adds a ran-
domly initialized BiLSTM on top of the “LTR + No
NSP” model during fine-tuning. ablation studies can be found in Appendix C. 5.1 E
We demonstrate the importance of the deep bidi-
rectionality of BERT by evaluating two pre-
training objectives using exactly the same pre-
training data, ﬁne-tuning scheme, and hyperpa-
rameters as BERT BASE :
No NSP : A bidirectional model which is trained
using the “masked LM” (MLM) but without the
“next sentence prediction” (NSP) task. LTR & No NSP : A left-context-only model
is trained using a standard Left-to-Right (LTR)
LM,
```



Tasks	Dev Set MNLI-m (Acc)	QNLI (Acc)	MRPC (Acc)	SST-2 (Acc)	SQuAD (F1)
$\text{BERT}_{\text{BASE}}$	84.4	88.4	86.7	92.7	88.5
No NSP	83.9	84.9	86.5	92.6	87.9
LTR & No NSP	82.1	84.3	77.5	92.1	77.8
+ BiLSTM	82.1	84.1	75.7	91.6	84.9

Table 5: Ablation over the pre-training tasks using the $\text{BERT}_{\text{BASE}}$ architecture. “No NSP” is trained without the next sentence prediction task. “LTR & No NSP” is trained as a left-to-right LM without the next sentence prediction, like OpenAI GPT. “+ BiLSTM” adds a randomly initialized BiLSTM on top of the “LTR + No NSP” model during fine-tuning.

ablation studies can be found in Appendix C.

5.1 Effect of Pre-training Tasks

We demonstrate the importance of the deep bidirectionality of BERT by evaluating two pre-training objectives using exactly the same pre-training data, fine-tuning scheme, and hyperparameters as $\text{BERT}_{\text{BASE}}$:

No NSP: A bidirectional model which is trained using the “masked LM” (MLM) but without the “next sentence prediction” (NSP) task.
LTR & No NSP: A left-context-only model which is trained using a standard Left-to-Right (LTR) LM, rather than an MLM. The left-only constraint was also applied at fine-tuning, because removing it introduced a pre-train/fine-tune mismatch that degraded downstream performance. Additionally, this model was pre-trained without the NSP task. This is directly comparable to OpenAI GPT, but using our larger training dataset, our input representation, and our fine-tuning scheme.

We first examine the impact brought by the NSP task. In Table 5, we show that removing NSP hurts performance significantly on QNLI, MNLI, and SQuAD 1.1. Next, we evaluate the impact of training bidirectional representations by comparing “No NSP” to “LTR & No NSP”. The LTR model performs worse than the MLM model on all tasks, with large drops on MRPC and SQuAD.

For SQuAD it is intuitively clear that a LTR model will perform poorly at token predictions, since the token-level hidden states have no right-side context. In order to make a good faith attempt at strengthening the LTR system, we added a randomly initialized BiLSTM on top. This does significantly improve results on SQuAD, but the results are still far worse than those of the pre-trained bidirectional models. The BiLSTM hurts performance on the GLUE tasks.

We recognize that it would also be possible to train separate LTR and RTL models and represent each token as the concatenation of the two models, as ELMo does. However: (a) this is twice as expensive as a single bidirectional model; (b) this is non-intuitive for tasks like QA, since the RTL model would not be able to condition the answer on the question; (c) this it is strictly less powerful than a deep bidirectional model, since it can use both left and right context at every layer.

5.2 Effect of Model Size

In this section, we explore the effect of model size on fine-tuning task accuracy. We trained a number of BERT models with a differing number of layers, hidden units, and attention heads, while otherwise using the same hyperparameters and training procedure as described previously.

Results on selected GLUE tasks are shown in Table 6. In this table, we report the average Dev Set accuracy from 5 random restarts of fine-tuning. We can see that larger models lead to a strict accuracy improvement across all four datasets, even for MRPC which only has 3,600 labeled training examples, and is substantially different from the pre-training tasks. It is also perhaps surprising that we are able to achieve such significant improvements on top of models which are already quite large relative to the existing literature. For example, the largest Transformer explored in Vaswani et al. (2017) is (L=6, H=1024, A=16) with 100M parameters for the encoder, and the largest Transformer we have found in the literature is (L=64, H=512, A=2) with 235M parameters (Al-Rfou et al., 2018). By contrast, $\text{BERT}_{\text{BASE}}$ contains 110M parameters and $\text{BERT}_{\text{LARGE}}$ contains 340M parameters.

It has long been known that increasing the model size will lead to continual improvements on large-scale tasks such as machine translation and language modeling, which is demonstrated by the LM perplexity of held-out training data shown in Table 6. However, we believe that this is the first work to demonstrate convincingly that scaling to extreme model sizes also leads to large improvements on very small scale tasks, provided that the model has been sufficiently pre-trained. Peters et al. (2018b) presented

Instantiate the Vectorstore

```
hain_community.vectorstores import Chroma
hunk_vectorstore = Chroma.from_documents(semantic_chunks, embedding=embed_model)
```

We will “limit” our semantic retriever to k = 1 to demonstrate the power of the semantic chunking strategy while maintaining similar token counts between the semantic and naive retrieved context.

Instantiate Retrieval Step

```
semantic_chunk_retriever = semantic_chunk_vectorstore.as_retriever(search_kwargs
semantic_chunk_retriever.invoke("Describe the Feature-based Approach with BERT?"

########################RESPONSE###################################
[Document(page_content='The right part of the paper represents the\nDev set resu
```

Instantiate Augmentation Step(for content Augmentation)

```
from langchain_core.prompts import ChatPromptTemplate

rag_template = """\
Use the following context to answer the user's query. If you cannot answer, plea

User's Query:
{question}

Context:
{context}
"""

rag_prompt = ChatPromptTemplate.from_template(rag_template)
```

Instantiate the Generation Step

```
chat_model = ChatGroq(temperature=0,
                      model_name="mixtral-8x7b-32768",
                      api_key=userdata.get("GROQ_API_KEY"),)
```

Creating a RAG Pipeline Utilizing Semantic Chunking

```
from langchain_core.runnables import RunnablePassthrough
from langchain_core.output_parsers import StrOutputParser

semantic_rag_chain = (
    {"context" : semantic_chunk_retriever, "question" : RunnablePassthrough()}
    | rag_prompt
    | chat_model
    | StrOutputParser()
)
```

Ask Question 1

```
semantic_rag_chain.invoke("Describe the Feature-based Approach with BERT?")

################ RESPONSE ###################################
The feature-based approach with BERT, as mentioned in the context, involves usin

To use BERT in a feature-based approach, the last 4 layers of BERT are concatena

The context also mentions that fine-tuning BERT is surprisingly robust to differ

In summary, the feature-based approach with BERT involves using the last 4 layer
```

Ask Question 2

```
semantic_rag_chain.invoke("What is SQuADv2.0?")
################ RESPONSE ###################################
SQuAD v2.0, or Squad Two Point Zero, is a version of the Stanford Question Answe
```

Ask Question 3

```
semantic_rag_chain.invoke("What is the purpose of Ablation Studies?")
################ RESPONSE ###################################
Ablation studies are used to understand the impact of different components or se
```

Implement a RAG pipeline using Naive Chunking Strategy

```
naive_chunk_vectorstore = Chroma.from_documents(naive_chunks, embedding=embed_mo
naive_chunk_retriever = naive_chunk_vectorstore.as_retriever(search_kwargs={"k"
naive_rag_chain = (
    {"context" : naive_chunk_retriever, "question" : RunnablePassthrough()}
    | rag_prompt
    | chat_model
    | StrOutputParser()
)
```

Note : Here we are going to use k = 5 ;this is to “make it a fair comparison” between the two strategies.

Ask Question 1

```
naive_rag_chain.invoke("Describe the Feature-based Approach with BERT?")

#############################RESPONSE#########################
The Feature-based Approach with BERT involves extracting fixed features from the
```

Ask Question 2

```
naive_rag_chain.invoke("What is SQuADv2.0?")
#############################RESPONSE#########################
SQuAD v2.0, or the Stanford Question Answering Dataset version 2.0, is a collect
```

Ask Question 3

```
naive_rag_chain.invoke("What is the purpose of Ablation Studies?")

#############################RESPONSE#########################
Ablation studies are used to evaluate the effect of different components or sett
```

```
For instance, one ablation study investigates the effect of the number of traini

Another ablation study focuses on different masking procedures during pre-traini
```

Ragas Assessment Comparison for Semantic Chunker

split documents using RecursiveCharacterTextSplitter

```
synthetic_data_splitter = RecursiveCharacterTextSplitter(
    chunk_size=1000,
    chunk_overlap=0,
    length_function=len,
    is_separator_regex=False
)
#
synthetic_data_chunks = synthetic_data_splitter.create_documents([d.page_content
print(len(synthetic_data_chunks))
```

Create the Following Datasets

- *Questions — synthetically generated (grogq-mixtral-8x7b-32768)*
- *Contexts — created above(Synthetic data chunks)*
- *Ground Truths — synthetically generated (grogq-mixtral-8x7b-32768)*
- *Answers — generated from our Semantic RAG Chain*

```
questions = []
ground_truths_semantic = []
contexts = []
answers = []

question_prompt = """\
You are a teacher preparing a test. Please create a question that can be answere

Context:
{context}
"""

question_prompt = ChatPromptTemplate.from_template(question_prompt)

ground_truth_prompt = """\
Use the following context and question to answer this question using *only* the
```

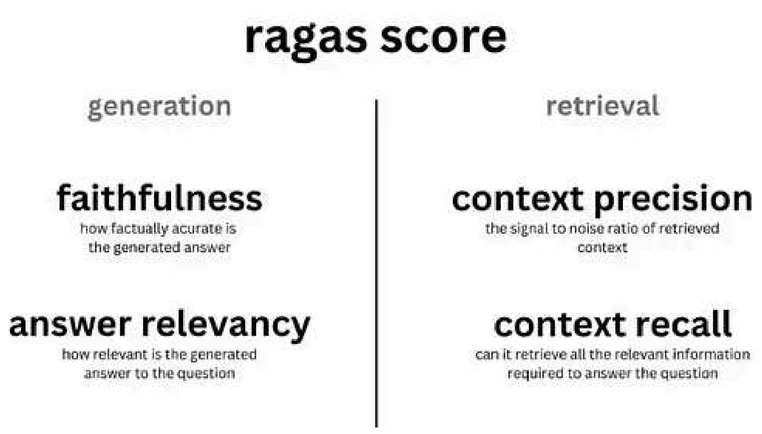
```
Question:
{question}

Context:
{context}
"""

ground_truth_prompt = ChatPromptTemplate.from_template(ground_truth_prompt)

question_chain = question_prompt | chat_model | StrOutputParser()
ground_truth_chain = ground_truth_prompt | chat_model | StrOutputParser()

for chunk in synthetic_data_chunks[10:20]:
  questions.append(question_chain.invoke({"context" : chunk.page_content}))
  contexts.append([chunk.page_content])
  ground_truths_semantic.append(ground_truth_chain.invoke({"question" : question
  answers.append(semantic_rag_chain.invoke(questions[-1]))
```

Note: for experimental purpose we have considered only 10 samples

Format the content generated into HuggingFace Dataset Format

```
from datasets import load_dataset, Dataset

qagc_list = []

for question, answer, context, ground_truth in zip(questions, answers, contexts,
  qagc_list.append({
      "question" : question,
      "answer" : answer,
      "contexts" : context,
      "ground_truth" : ground_truth
  })

eval_dataset = Dataset.from_list(qagc_list)
eval_dataset

###########################RESPONSE###########################
Dataset({
    features: ['question', 'answer', 'contexts', 'ground_truth'],
    num_rows: 10
})
```

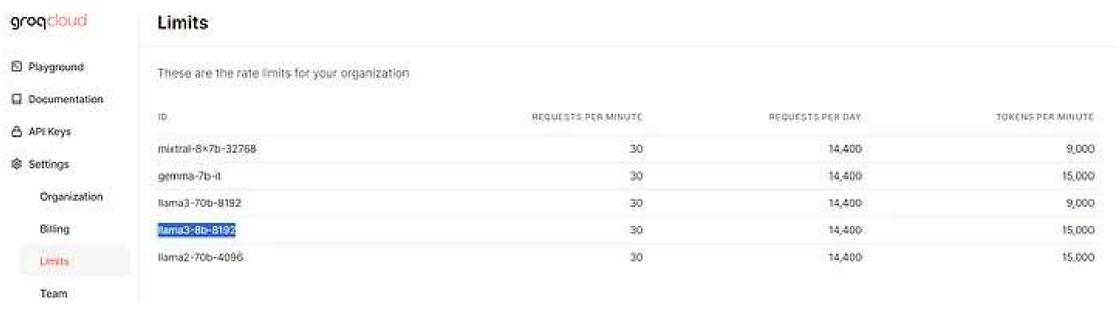
Implement Ragas metrics and evaluate our created dataset.


ragas score
generation
retrieval
faithfulness
how factually acurate is the generated answer
answer relevancy
how relevant is the generated answer to the question
context precision
the signal to noise ratio of retrieved context
context recall
can it retrieve all the relevant information required to answer the question

```
from ragas.metrics import (
    answer_relevancy,
    faithfulness,
    context_recall,
    context_precision,
)

#
from ragas import evaluate

result = evaluate(
    eval_dataset,
    metrics=[
        context_precision,
        faithfulness,
        answer_relevancy,
        context_recall,
    ],
     llm=chat_model,
    embeddings=embed_model,
    raise_exceptions=False
)
```

Here I had tried to use open source LLM using Groq. But got a rate limit error :

```
limits for more information.', 'type': 'tokens', 'code': 'rate_limit_exceeded'}}
```



Limits

These are the rate limits for your organization

ID	REQUESTS PER MINUTE	REQUESTS PER DAY	TOKENS PER MINUTE
mixtral-8x7b-32768	30	14,400	9,000
gemma-7b-it	30	14,400	15,000
llama3-70b-8192	30	14,400	9,000
llama3-8b-8192	30	14,400	15,000
llama2-70b-4096	30	14,400	15,000

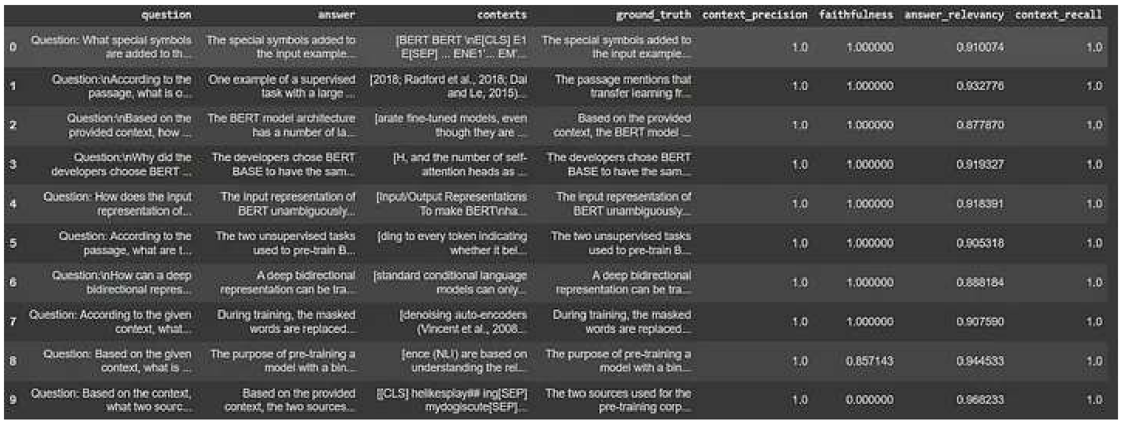
So redirected the LLM to use OpenAI which is by default used in RAGAS framework.

Set up OpenAI API keys

```
import os
from google.colab import userdata
import openai
os.environ['OPENAI_API_KEY'] = userdata.get('OPENAI_API_KEY')
openai.api_key = os.environ['OPENAI_API_KEY']
```

```
from ragas import evaluate

result = evaluate(
    eval_dataset,
    metrics=[
        context_precision,
        faithfulness,
        answer_relevancy,
        context_recall,
    ],
)
result

#########################RESPONSE##########################
{'context_precision': 1.0000, 'faithfulness': 0.8857, 'answer_relevancy': 0.9172
```

```
#Extract the details into a dataframe
results_df = result.to_pandas()
results_df
```



	question	answer	contexts	ground_truth	context_precision	faithfulness	answer_relevancy	context_recall
0	Question: What special symbols are added to th...	The special symbols added to the input example...	[BERT BERT \nE[CLS] E1 E[SEP] ... ENE1' ... EM'...	The special symbols added to the input example...	1.0	1.000000	0.910074	1.0
1	Question:\nAccording to the passage, what is o...	One example of a supervised task with a large ...	[2018; Radford et al., 2018; Dai and Le, 2015)...	The passage mentions that transfer learning fr...	1.0	1.000000	0.932776	1.0
2	Question:\nBased on the provided context, how ...	The BERT model architecture has a number of la...	[arate fine-tuned models, even though they are ...	Based on the provided context, the BERT model ...	1.0	1.000000	0.877870	1.0
3	Question:\nWhy did the developers choose BERT ...	The developers chose BERT BASE to have the sam...	[H, and the number of self-attention heads as ...	The developers chose BERT BASE to have the sam...	1.0	1.000000	0.919327	1.0
4	Question: How does the input representation of...	The input representation of BERT unambiguously...	[Input/Output Representations To make BERT\nha...	The input representation of BERT unambiguously...	1.0	1.000000	0.918391	1.0
5	Question: According to the passage, what are t...	The two unsupervised tasks used to pre-train B...	[ding to every token indicating whether it bel...	The two unsupervised tasks used to pre-train B...	1.0	1.000000	0.905318	1.0
6	Question:\nHow can a deep bidirectional repres...	A deep bidirectional representation can be tra...	[standard conditional language models can only...	A deep bidirectional representation can be tra...	1.0	1.000000	0.888184	1.0
7	Question: According to the given context, what...	During training, the masked words are replaced...	[denoising auto-encoders (Vincent et al., 2008...	During training, the masked words are replaced...	1.0	1.000000	0.907590	1.0
8	Question: Based on the given context, what is ...	The purpose of pre-training a model with a bin...	[ence (NLI) are based on understanding the rel...	The purpose of pre-training a model with a bin...	1.0	0.857143	0.944533	1.0
9	Question: Based on the context, what two sourc...	Based on the provided context, the two sources...	[[CLS] helikesplay## ing[SEP] mydogiscute[SEP]...	The two sources used for the pre-training corp...	1.0	0.000000	0.968233	1.0

Ragas Assessment Comparison for Naive Chunker

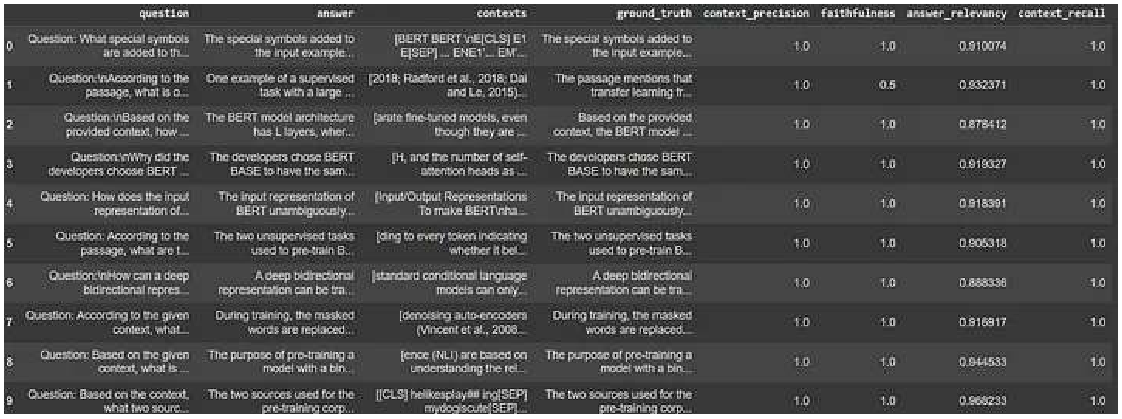
```
import tqdm
questions = []
ground_truths_semantic = []
contexts = []
answers = []
for chunk in tqdm.tqdm(synthetic_data_chunks[10:20]):
  questions.append(question_chain.invoke({"context" : chunk.page_content}))
  contexts.append([chunk.page_content])
  ground_truths_semantic.append(ground_truth_chain.invoke({"question" : question
  answers.append(naive_rag_chain.invoke(questions[-1]))
```

Formulate naive chunking evaluation dataset

```
qagc_list = []

for question, answer, context, ground_truth in zip(questions, answers, contexts,
  qagc_list.append({
      "question" : question,
      "answer" : answer,
      "contexts" : context,
      "ground_truth" : ground_truth
  })

naive_eval_dataset = Dataset.from_list(qagc_list)
naive_eval_dataset

############################RESPONSE########################
Dataset({
    features: ['question', 'answer', 'contexts', 'ground_truth'],
        num_rows: 10
    })
```

Evaluate our created dataset using RAGAS framework

```
sult.to_pandas()


####RESPONSE ########################
0, 'faithfulness': 0.9500, 'answer_relevancy': 0.9182, 'context_recall': 1.0000}
```



	question	answer	contexts	ground_truth	context_precision	faithfulness	answer_relevancy	context_recall
0	Question: What special symbols are added to th...	The special symbols added to the input example...	[BERT BERT \nE[CLS] E1 E[SEP] ... ENE1' ... EM'...	The special symbols added to the input example...	1.0	1.0	0.910074	1.0
1	Question:\nAccording to the passage, what is o...	One example of a supervised task with a large ...	[2018; Radford et al., 2018; Dai and Le, 2015)...	The passage mentions that transfer learning fr...	1.0	0.5	0.932371	1.0
2	Question:\nBased on the provided context, how ...	The BERT model architecture has L layers, wher...	[arate fine-tuned models, even though they are ...	Based on the provided context, the BERT model ...	1.0	1.0	0.878412	1.0
3	Question:\nWhy did the developers choose BERT ...	The developers chose BERT BASE to have the sam...	[H, and the number of self-attention heads as ...	The developers chose BERT BASE to have the sam...	1.0	1.0	0.918327	1.0
4	Question: How does the input representation of...	The input representation of BERT unambiguously...	[Input/Output Representations To make BERT\nha...	The input representation of BERT unambiguously...	1.0	1.0	0.918391	1.0
5	Question: According to the passage, what are t...	The two unsupervised tasks used to pre-train B...	[ding to every token indicating whether it bel...	The two unsupervised tasks used to pre-train B...	1.0	1.0	0.905318	1.0
6	Question:\nHow can a deep bidirectional repres...	A deep bidirectional representation can be tra...	[standard conditional language models can only...	A deep bidirectional representation can be tra...	1.0	1.0	0.888336	1.0
7	Question: According to the given context, what...	During training, the masked words are replaced...	[denoising auto-encoders (Vincent et al., 2008...	During training, the masked words are replaced...	1.0	1.0	0.916917	1.0
8	Question: Based on the given context, what is ...	The purpose of pre-training a model with a bin...	[ence (NLI) are based on understanding the rel...	The purpose of pre-training a model with a bin...	1.0	1.0	0.944533	1.0
9	Question: Based on the context, what two sourc...	The two sources used for the pre-training corp...	[[CLS] helikesplay## ing[SEP] mydogiscute[SEP]...	The two sources used for the pre-training corp...	1.0	1.0	0.968233	1.0

Conclusion

Here we can see that the results of both Semantic Chunking and Naive Chunking are almost same except that Naive Chunker has a better factual representation of answer with a score of 0.95 when compared to score of 0.88 of Semantic Chunker.

In conclusion, semantic chunking enables the grouping of contextually similar information, allowing for the creation of independent and meaningful segments. This approach enhances the efficiency and effectiveness of large language models by providing them with focused inputs, ultimately improving their ability to comprehend and process natural language data.

References:

GroqCloud
Experience the fastest inference in the world
console.groq.com

Metrics | Ragas
Skip to content Just like in any machine learning system, the performance of individual components within the LLM and…
docs.ragas.io

Semantic Chunking | 🦜🔗 LangChain
Splits the text based on semantic similarity.
python.langchain.com

Semantic Chunking Langchain

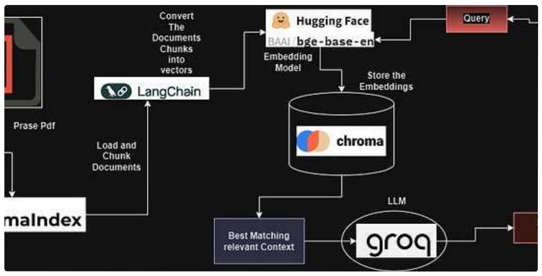
Written by Plaban Nayak

Following

1.93K Followers · Editor for The AI Forum

Machine Learning and Deep Learning enthusiast

More from Plaban Nayak and The AI Forum



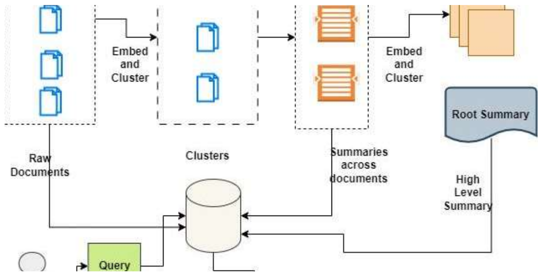
Plaban Nayak in The AI Forum

RAG on Complex PDF using LlamaParse, Langchain and Groq

Retrieval-Augmented Generation (RAG) is a new approach that leverages Large Languag...

13 min read · Apr 7, 2024

357 3



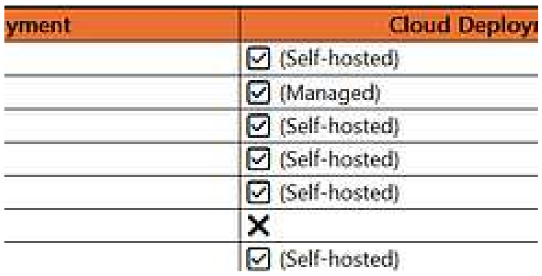
Plaban Nayak in The AI Forum

Implementing Advanced RAG in Langchain using RAPTOR

Usually in conventional RAG we often rely on retrieving short contiguous text chunks for...

283 min read · Mar 24, 2024

316 1



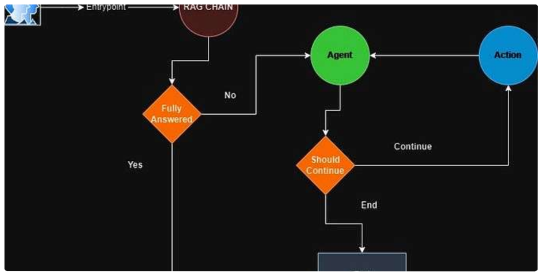
yment	Cloud Deploy
	☑ (Self-hosted)
	☑ (Managed)
	☑ (Self-hosted)
	☑ (Self-hosted)
	☑ (Self-hosted)
	✗
	☑ (Self-hosted)

Plaban Nayak in The AI Forum

Which Vector Database Should You Use? Choosing the Best One for...

Introduction

9 min read · Apr 19, 2024



Plaban Nayak in The AI Forum

Implementing Agentic RAG using Langchain

Native RAG

12 min read · Mar 31, 2024

264 3 237

See all from Plaban Nayak

See all from The AI Forum

Recommended from Medium

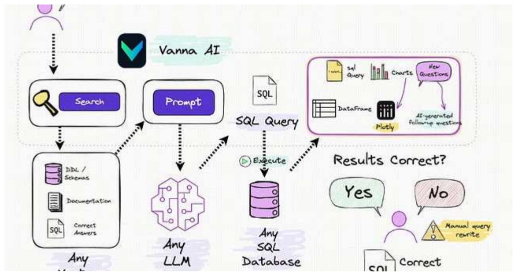
Aniket Hinga... in Artificial Intelligence in Plain En...

RAG 2.0 : Your AI's Scattered Brain Just Got Organized

Teamwork Makes the Dream Work: The Collaborative Core of RAG 2.0

· 9 min read · Apr 23, 2024

319 1



Angelina Yang

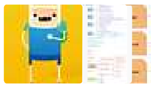
No More Text2SQL, It's Now RAG2SQL!

Text2SQL was awfully popular last year. I talked with a vendor looking to buy, and also...

· 3 min read · Apr 13, 2024

149 3

Lists

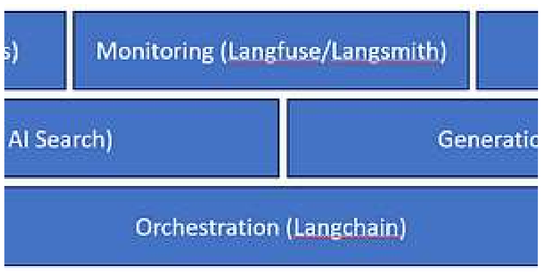


Natural Language Processing

1419 stories · 917 saves

Staff Picks

634 stories · 934 saves



Anurag Mishra in CodeX

Build End-to-End RAG Pipeline with Monitoring and Evaluation using…

Introduction

7 min read · Apr 21, 2024

109

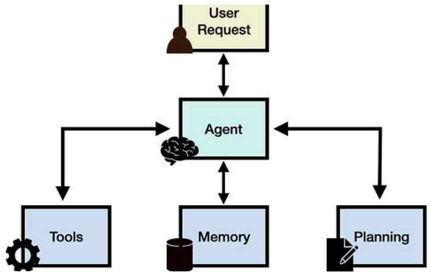
Silvia Onofrei in Towards Data Science

Relation Extraction with Llama3 Models

Enhanced relation extraction by fine-tuning Llama3–8B with a synthetic dataset created…

12 min read · Apr 26, 2024

424 2



Vishal Rajput in AIGuys

AI Agents Are All You Need

AI Agents, Understanding the role of Tools, Memory, and Planning in making them work.

· 14 min read · Apr 22, 2024

447 1

Fabio Matricardi in The AI Explorer

Microsoft strikes back—Phi-3 may change the entire game.

A Highly Capable Language Model Locally on Your Phone! We have already the GGUF…

· 9 min read · Apr 24, 2024

614 3

See more recommendations