Title: A Comparison of Dynamic Chunk Size Methods

Authors:

Jonathan Roig[1], Laura Dozal[2], Melissa J. Evans[3],and Lane Harrison[4]

1 US Government; jonathan.roig@outlook.com
2 University of Arizona; lwerthmann@arizona.edu
3 Vanderbilt University; melissa.j.evans@vanderbilt.edu
4 Worcester Polytechnic Institute; ltharrison@wpi.edu
*contact email: Jonathan Roig: jonathan.roig@outlook.com

Abstract:

Sentences, paragraphs, and documents are often used as the standard chunk size for text embedding, but are any the optimal size for summarization and RAG tasks? We address this question in three parts: the creation of a summarization UI that allows users to compare the quality of summaries side-by-side with different chunk size methods on the backend, as well as the quantitative ROUGE scores compared to ground truth summaries; a simulation of RAG evaluation with vector databases created from different chunk size methods, evaluated with RAGAs; and a task given to analysts asking them to break down text into chunks of information to determine if any patterns might suggest a novel chunk size method. We find that most chunk size methods do not significantly influence the output of summarization using `occams`, but overlapping 10 sentence chunks perform worse than others. We find that paragraph and semantic methods perform similarly for RAG, but the recursive method performs worse. Finally, we conclude that analysts widely vary in their perception of information chunks.

Keywords:

Text embeddings, RAG, survey, tailored summaries, unit of information

Github Repository:

https://github.ncsu.edu/SCADS/chunk-size

1. Introduction

Chunk size for embeddings has become a renewed interest after the rise of Large Language Models (LLMs) and especially after the addition of Retrieval Augmented Generation (RAG). Sentences, paragraphs, and documents are often used as the standard embedding unit size, but are they the optimal size for summarization and RAG tasks? Can we make chunks small enough to restrict unnecessary information and large enough to include necessary context? Can analyst perceptions of information segmentation be a proxy for a good chunk size. This project seeks to compare a variety of chunk size methods via their effects on the output of summarization. Our goal is to determine the optimal chunk-sizing process for embedding documents for summarization and RAG by focusing on chunk size methods that do not use a consistent grammatical pattern ("dynamic"), and therefore use variable lengths. We have split the project into three overarching efforts: 1) creating a testing environment with which users can test different types of summarization based on varied document types, encoder models, and chunk sizes; 2) evaluating the effects of chunk size methods on the creation of vector databases in RAG, and; 3) finding an analyst-informed structure of an information unit through the use of a survey task.

Summarization Testing Environment for Various Chunk Size Methods

The testing environment is a user interface that allows for both qualitative and quantitative evaluation of summarization. Users are able to compare summaries side-by-side as well as see their ROUGE scores when compared to a ground truth summary. The original document is also available. The UI includes the ability to choose from among multiple datasets, summarizers, text embedders, and chunk size methods. Users look through the results for a single document at a time.

In the future, the UI will also implement RAG with its own chunk size summary review, but for this research we made sure to first test and compare chunk sizes on the back end. Here our goal is to identify which chunking mechanism provides the right amount of context to both prevent hallucinations and avoid missing any crucial entities. Testing was implemented using TREC MSMarco datasets and synthetic question and answering data created from those datasets. A jupyter notebook on merging the datasets and implementing the RAG chunk size comparisons can be found in the GitHub repository.

A second part of this project begins to investigate the idea of a "unit of information" specifically from the perspective of an analyst. Here we introduced a chunking task where analysts were asked to segment unformatted text documents about different topics based on where they thought a new idea, or a new unit of information was present. The hope is to use an analyst's idea of the unit of information as an embedding unit to inform our chunk size methods that will lead to optimal outputs. As we have seen in other projects at SCADS, analysts prefer to have choices based on the mission they are given. Thus we hoped this survey would let us see the unique style in which an analyst may process documents that can not be intuited by looking for grammatical patterns or following traditional chunking conventions.

2. Background

2.1 Text segmentation (Chunking)

Text segmentation, also known as chunking, refers to strategies used to break down documents of text into smaller pieces, usually using some heuristic related to length, grammatical structure, semantic relation, or a combination of these. A version of this idea is the context window seen in LLMs. This is usually a static number of tokens that LLMs are limited to for input. This represents one of the most static methods by imposing a length that has little or nothing to do with grammatical structure or semantic meaning, depending on the LLM's definition of "tokens". Other common methods take a grammatical structure, like the paragraph, as a chunk size. While this is still a relatively consistent pattern, paragraph lengths do vary. A style of chunk size method that uses more consistency is chunk overlap. This is when some of each chunk contains the same information as an adjacent chunk to prevent loss of information at the beginning and end of the chunk. For example, a chunk might be 10 sentences long, but the first 5 sentences match the last 5 sentences of the previous chunk. On the other end of the spectrum, some chunk size methods attempt to identify semantic groupings of text to create chunks. These include LangChain's semantic chunker, which uses text embeddings to group sentences together into cohesive chunks, or document-aware chunking, which uses the structure of documents to find groups of topically-related information.

The benefit of chunk size methods that go beyond token count and grammatical structure is that pulling chunks together by using semantic groupings may allow systems to retrieve and summarize the proper amount of information related to what a user is looking for. For example, if a user is using a system that chunks entire documents, but only needs information that a single paragraph would be enough to provide, they may receive lots of unnecessary context that they have to filter through to find their information. On the other hand, if we chunk at the sentence level, users may miss out on related context needed to understand the information they receive. However, static chunk size methods gain the benefit of efficiency as they can use regular expressions, which run much faster than semantic methods that rely on text embeddings and/or LLMs by bypassing the use of neural methods. A good semantic chunking method must be worth the complexity trade-off.

2.2 Intelligence Analysts

In the field of intelligence analysis, analysts have a unique workflow of trying to process innumerate quantities of data to try and discern information. As new AI technology is on the rise, many are asking if AI tools can be used to help facilitate this challenging workflow. The premise of SCADS, where this work was conducted, is to create a tailored daily report (TLDR) for intelligence analysts that is AI generated to help summarize the state of affairs before they dive into their work. Thu,s LLMs will be heavily relied upon to generate summaries of the many documents available to analysts.

There remain two relevant problems with this aim of SCADS that the work in this paper attempts to tackle - first, what analysts need summaries for is different than what most LLMs are being built for. Analysts are detail oriented people who need to try and find connecting themes

and threads across documents, and thus many want AI-generated summaries as a tool to help them figure out *which* documents to look at, not to replace having to read the full document. Second, analysts are incredibly unique in their domain expertise and workflow habits, thus the ideal summary for one analyst will never be the same as the ideal summary for another. Thus the ideal TLDR is truly Tailored to each analyst. This is often conceptualized as tailoring which articles get recommended, but it should also be considered in how those articles are then summarized to match the workflow and cognitive processes of each analyst.

3. Methodology

3.1 Chunk Size Methods

We implemented a number of methods for text segmentation and used two from LangChain packages. While we were not able to apply all to both summarization and RAG, each of these is represented in at least one part of the work.

Sentence

Sentence chunking simply breaks documents into its sentences. We use NLTK's sentence tokenizer, which uses their PunktSentenceTokenizer behind the scenes. This algorithm uses an unsupervised algorithm to understand distinctions between sentence boundaries and abbreviations. This is one step above the naive method of splitting on punctuation, which inevitably splits some sentences incorrectly by splitting on other uses of punctuation that are not sentence boundaries (e.g. "Mr.").

Paragraph

Paragraph chunking breaks documents into paragraphs. This is done using a regular expression that splits on any number of new lines. This is imperfect, as it will capture anything that exists as its own line as a paragraph, to include section headings; however, in most cases, this method captures the general idea of a paragraph. The benefit is that such a regular expression runs efficiently.

Document

Document chunking keeps documents in their entirety. No changes are made, and length depends entirely on the lengths of documents themselves. This is the most efficient as it requires no preprocessing, but may lead to information dilution.

Overlapping Sentences

Our implementation of overlapping sentences creates 10 sentence chunks, where the first 5 sentences overlap with the previous chunk, and the last 5 sentences overlap with the next chunk, creating a sliding window effect. The only sentences that are not represented in two chunks are the first 5 and the last 5. The intended benefit is to avoid losing information due to the effects of separating information that may be cohesive.

Semantic

Semantic chunking attempts to condense sentences with high semantic similarity into a single chunk. This method still begins with sentence size chunks. All sentences are embedded, then compared to all other sentences to find sentences that meet the threshold for similarity. Those that do are merged together. The method also has an option of considering sentences with a buffer of adjacent sentences, similar to chunk overlap. This is the only method we consider that tries to account for semantic cohesion, with the expense of additional text embeddings and similarity comparisons.

Recursive (Naive Chunking)
Recursive chunking divides the input text into smaller chunks in a hierarchical and iterative manner based on an input set of separators. If the first separator doesn't result in the expected chunk size or structure, the method will recursively be implemented with the next separator to split the text until it outputs the correct chunk size or structure asked. Here, chunks won't be exactly the same size, but they will be close because they leverage fixed chunk sizes and overlap processes. This method gains the benefit of paying attention to structure while maintaining relative consistency.

Multilevel
This is a novel method we propose that includes multiple methods of static embedding embedded into the same space. All text is represented three times: once each in a sentence, paragraph, and document-sized chunk. The intended benefit is that this allows a query to determine which chunk size it is most similar to, ideally pulling back the one that includes the proper amount of context for the user's needs. While this is primarily a potential RAG benefit, we started by implementing it in our summarization UI.

3.2 Summarization Testing Environment to Compare Chunk Size Methods

We created a user-centric testing environment designed to allow users to explore the effects of chunk size on a summarization pipeline. Users are provided with the opportunity to make their own qualitative judgments on the summaries, but are also provided with the results of ROUGE score testing for datasets that have ground truth summary data.

3.2.1 Summarization pipeline

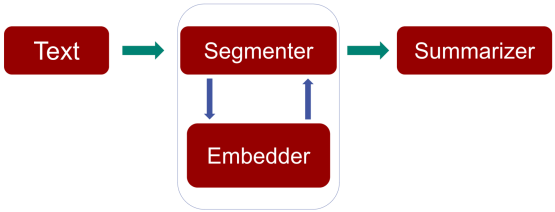
Our summarization pipeline used `occams`, an extractive summarizer. This summarizer uses an optimization algorithm to produce extractive summarizers from provided text. Importantly, it is able to perform multi-document summarization, which allows us to test our chunks by treating them as individual documents. The summarizer does require a summary budget to be set. This is a maximum length for summaries, though `occams` uses as much of that length as possible, leading to summaries that are greatly the same length. Due to the length of articles in our primary dataset, we opted to set this budget at 100 words. Most important is that this number applies to all generated summaries, providing some level of normalization for evaluation.

The datasets available in the UI are the CNN/Daily Mail News dataset available on Kaggle, and the MS Marco documents dataset. The CNN/Daily Mail News dataset is the only of these that has ground truth summaries, allowing us to produce quantitative evaluations in addition to the user capability to evaluate qualitatively.

The chunk sizes used for the summarization pipeline are sentence, paragraph, document, overlapping sentences, semantic, and multilevel.

The embedder used for all text embeddings is the OpenAI text-embedding-ada-002 model.

The process for turning text into summaries is to first chunk the text (which may require text embeddings), then feed those chunks to the multidocument summarizer. This is done for each selected chunk size. Each summary is returned to the UI for display to the user.



3.2.2 User interface

The user interface was created using Streamlit, a Python package designed to assist in creating user interfaces, especially for AI/ML tasks. The core of the design is to provide users with options that allow them to explore the quality of summaries based on the parameters they choose, with the greatest emphasis on chunk size method.

The user interface first requires users to enter an OpenAI API Token. It will then by default load in the CNN/Daily Mail News dataset since this is the one with quantitative evaluation options. Dropdown menus are available for dataset, summarizer, and embedding model. The current version has two dataset options, but only one summarizer and embedding model at the time of writing. Three summaries are immediately visible: those made with paragraph, semantic, and multilevel chunking. Users are limited to three summaries at once to prevent information overload and cluttered text. Users are limited to a single document at once, as well, but can iterate freely through documents in the dataset, or enter the document ID of a specific document they may already be familiar with.

The ground truth summaries are visible on the UI for each document where available (in this case, the CNN/Daily Mail News dataset), as well as ROUGE-1 through 4 scores for each selected chunk size. For all datasets, the UI provides summaries of the article text for each selected chunk size as well as the original document text.

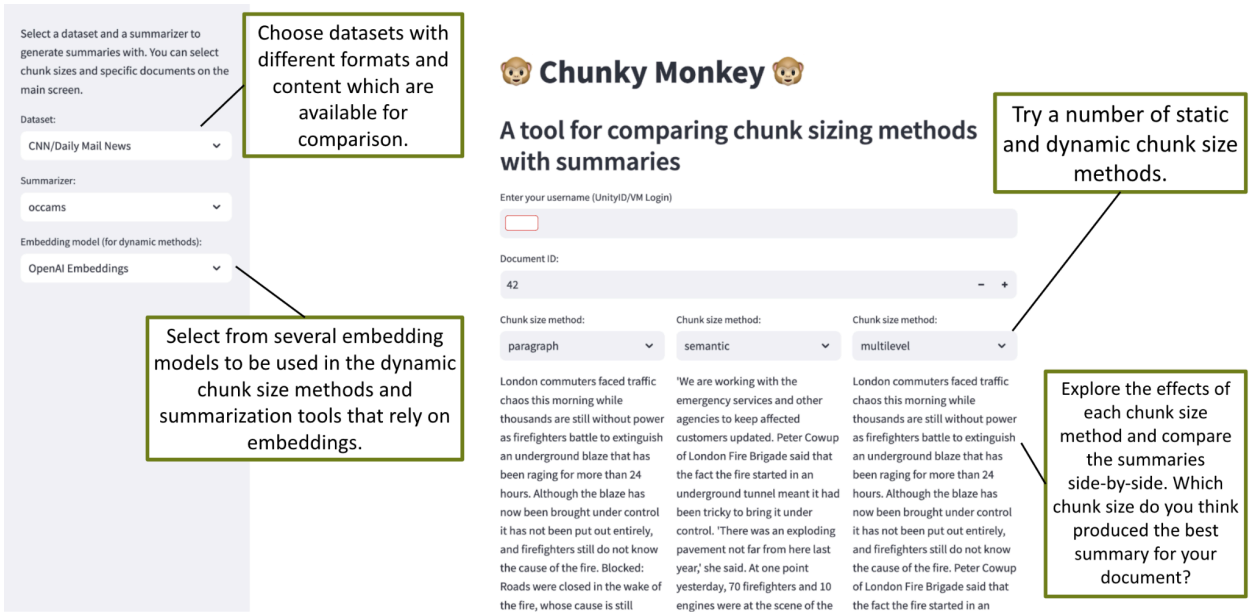


Figure x: The main screen of the UI, displaying the dropdowns for all of the selectable options and a portion of the summaries that might be generated from a document using different chunk sizes.

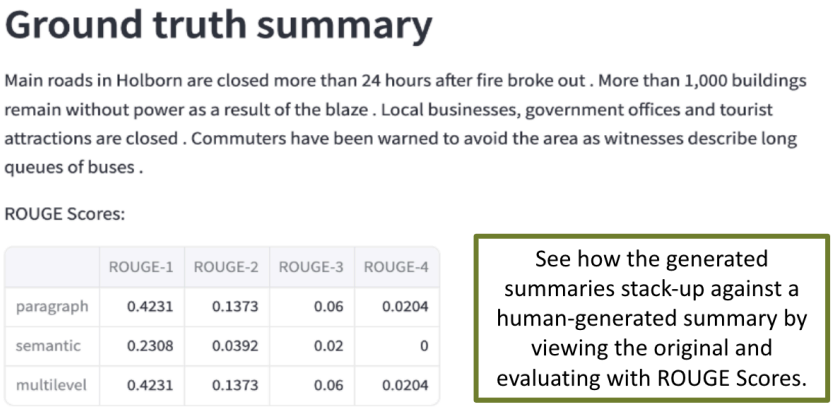


Figure y: An example of a ground truth summary with ROUGE scores for paragraph, semantic, and multilevel chunking.

3.2.3 Evaluation

In addition to the user's ability to judge individual documents qualitatively and observe the quantitative ROUGE score, we computed metrics for the entire CNN/Daily Mail News dataset using a jupyter notebook. The dataset contains 11,490 test documents with ground truth summaries.

We use the recall metric from ROUGE in keeping with its intention of being a recall metric. We specifically use ROUGE-1 for this overall evaluation, which looks at recall of unigrams between

the generated summary and the ground truth summary. We compare semantic, paragraph, overlapping sentences, and multilevel chunk size methods in this evaluation.

3.3 RAG implementation

RAG (Retrieval Augmented Generation) is an attempt to alleviate LLM hallucinations. LLM hallucination is the concept that LLMs generate wrong answers with the confidence to make the user believe it's true. This has been a major problem since the introduction of the LLMs. These hallucinations lead to incorrect and factually wrong answers. We implemented a RAG pipeline using various chunk size methods and evaluated these chunks using RAGAs evaluation metrics. This pipeline can be found in the RAGAs_comparison.ipynb notebook from the chunk-size repository.

The RAG Process

RAG segments a document text into chunks; based on the segmenter implemented. Then, it creates those chunks as embeddings (numerical representations) and stores them in a vector database. It then retrieves these embeddings to augment an LLM response. By doing this, the LLM uses the chunks as contextual support to generate an answer. In this project we tested the semantic chunker and paragraph chunk segments using a sample of 1000 documents of the MSMarco documents dataset.

Dataset

MSMarco Question Answering dataset version 2 (MSMarcoQA) and MSMarco Documents version 2 (MSMarcoD) datasets were used in the RAG pipeline. These datasets were combined on document urls to have ground truth, questions, answers, and context all in one place. For example, the MSMarcoQA holds the query, the query id, the query type, and segmented passages from the documents. The MSMarcoD holds the documents, their url, title, and headings. Here we implemented a chat prompt template to create synthetic questions that were used in conjunction with the various chunking implementations in RAG to output an answer. These answers were evaluated using the RAGAs metric.

Evaluating RAG

The evaluation is primarily quantitative, using RAGAs metrics for scoring our RAG answers. These metrics are scored on a merged dataset of the newly created chunks, MSMarco documents and a sample of 1000 documents of the MSMarco QA dataset. These metrics rely on answers, questions, contexts, and ground truth. Here, questions come from the QA dataset to be used with the RAG model; context comes from the various chunk implementations; answers come from a RAG pipeline that uses the different chunks; and ground truth in this case is pulled from the body of the sampled MSMarco documents to make sure the RAG pipeline gets that information.

Implementation of RAG was done by following tutorials using the Langchain and RAGAs documentation. The time only allotted for using synthetic testing data, but future implementation will look at the full dataset. The beginnings of this implementation are in the RAGAs comparison jupyter notebook.

3.4 Analyst Survey Chunking Task

We asked 5 analysts to examine a set of 4 documents and break them into information chunks. The goal of this task was to gather qualitative insight into how different analysts might process a document in terms of chunks. Then we could see if an embedding style that resembles these analyst behaviors could improve automatic text summarization. This survey was hosted online on reVISit, thus participants completed the survey on their own devices when they had time to do so (Ding et al., 2023).

The analysts who participated varied in self-reported experience (10-35 years experience, backgrounds include translation, signal intelligence, geopolitical information, network discovery, cybersecurity, and more). They read four documents: a technical paper excerpt about human activity detection via wireless signal (531 words), an intel report about a fictitious country's political turmoil (236 words), a news article about Hong Kong protests (380 words), and a cybersecurity report about ransomware (405 words). Documents came from the MIND dataset curated by Microsoft (Wu et al., 2020), Arxiv, and datasets created at SCADS.

Participants were given as much time as they desired to read a document and click on spaces to create "chunks". Once they did so, a blue mark appeared designating the chunk boundary they had just made. These chunk marks could be removed if they wished to change an answer. They could create a chunk anywhere there was a space in the text. We aimed to give minimal direction on what a chunk should look like. The instructions stated "Create a new chunk anytime you think new information is introduced in a text. Information chunks can be as specific or general as you want".

Additionally, on each document we asked participants two questions: "How often have you worked with documents structured like this?" and "How often have you worked with this kind of content?".To both questions they could select "Never", "Rarely", "Sometimes", or "Often". These questions were asked to help frame the chunks a participant chose to make by if that document was something they have experience processing. At the end of the survey we asked them to report on the process or approach they had for completing this task to further help us interpret their results.

4. Results

4.1 Summarization Results using ROUGE-1

The summarization task results were measured using ROUGE-1 scoring, the unigram variant of ROUGE, specifically using the recall metric. While the long-term goal of this project would be to include human evaluation, ROUGE scores can serve as a good initial measure as it is efficient and does not rely on LLMs or humans to evaluate.

Our results suggest that despite overall similar performance, the overlapping sentences method performs significantly worse than the semantic, paragraph, and multilevel embedding methods. We found this by applying a Kruskal-Wallis test to determine if any groups contained a significant difference from the others. With alpha = 0.05, we found a significant difference with a

$p < 0.0003$. To determine which groups this significant difference came from, we applied Tukey's honestly significant difference test. From this, we found that in each pair using the overlapping sentences method, there was a significant difference ($p < 0.0001$) in which the overlapping sentences method performed with a lower mean ROUGE-1 score than any of its counterparts. No other significant differences were found among semantic, paragraph, and multilevel chunking methods for summarization when measured with ROUGE-1.

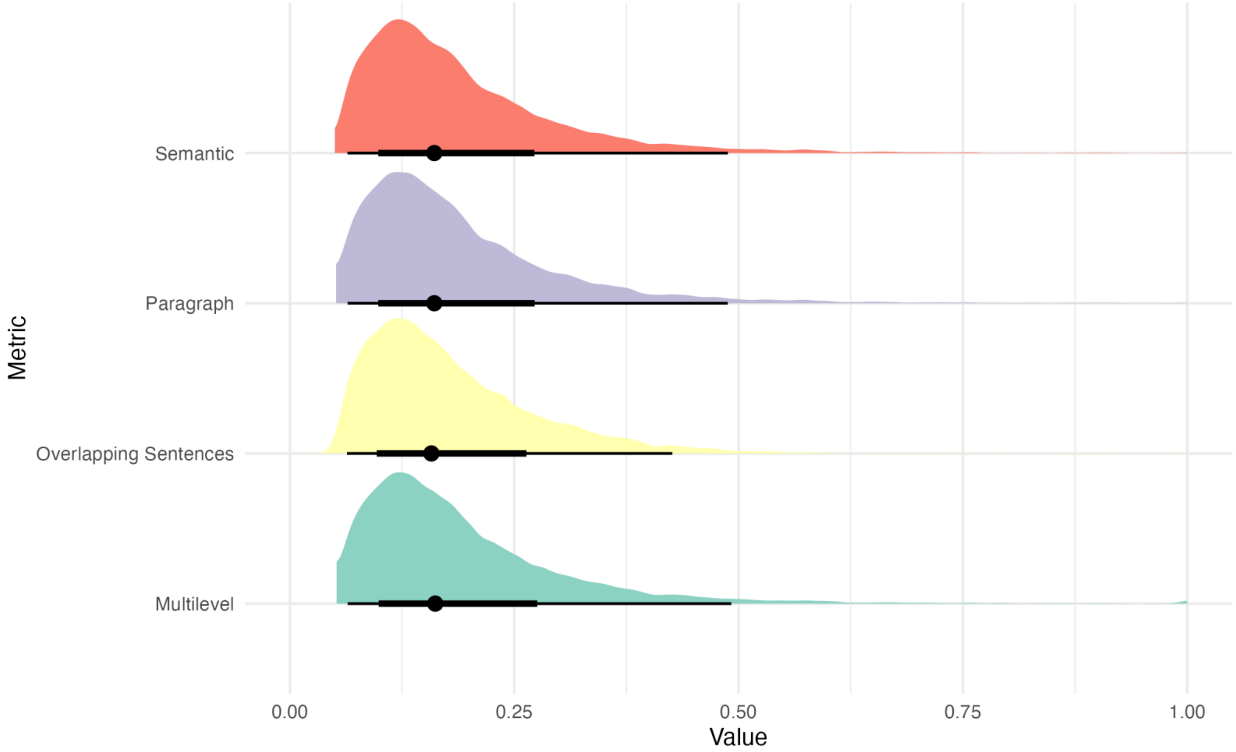


Figure x: The distribution of ROUGE-1 scores for each tested chunking method plotted with condensed box plots including a point for mean.

4.2 RAG Results Using RAGAs Evaluation Metrics

The evaluation metrics come from the RAGAs evaluator and can be split into the various parts of the RAG pipeline.

- Chunking Metrics
    - Context Recall: Measures how well the retrieved context aligns with the ground truth answer.
    - Context Entity recall: Looks at the probability that both context entities and ground truth entities are found in both, over the total number of ground truth entities.
        - Each sentence in the ground truth answer is analyzed to determine whether it can be attributed to the retrieved context or not. In an ideal scenario, all sentences in the ground truth answer should be attributable to the retrieved context.
- Embedding Metrics
    - Answer Correctness: Accuracy of generated answer compared to ground truth.

    - Answer semantic similarity: Assesses the semantic resemblance between the generated answer and the ground truth, I.e. alignment between the generated answer and ground truth.
- Overall Metrics
    - Faithfulness: Measures the factual consistency of the generated answer against the given context (information found in the document).
    - Answer Relevance: Measures how pertinent the generated answer is to the given prompt. Low score for incomplete answers or those that contain redundant information, high score for better relevancy.

RAGAs Evaluation Metrics Results

Chunking Method	Context Recall	Context Entity Recall	Answer Correctness	Answer Semantic Similarity	*Faithfulness*	Answer Relevance
Paragraph	0.9881	0.3265	0.6395	0.9088	0.8528	0.8004
Recursive	0.9500	0.3099	0.5889	0.8900	0.6254	0.7222
Semantic	0.9500	0.4756	0.6982	0.9237	0.8000	0.6580

Table 1*(?): Results for the RAGAs evaluation using synthetic Question and Answer data created from the MSMarco documents dataset.*

Chunking Evaluation: From Table 1, we see that the *context entity recall* is low throughout, meaning that the chunking implementations don't exactly split the text based on entities. In this case, there are more entities found in the ground truth data than in the chunks, and the semantic chunks score the highest (0.475). There is a high *context recall* between the ground truth data and the retrieved context from ground truth data. In this case, the paragraph chunking method scored the highest (0.988), recursive and semantic had no difference.

Embedding Metrics: By looking at how well the answer connects to the ground truth, and with what amount of factual similarity it does that, *answer correctness* (Table 1) in this case was higher in the semantic chunking method (0.698), showing that grouping of similar semantic topics is helpful here. While reviewing *answer semantic similarity* to find out how well the semantic output of the generated answer compares to the ground truth (Table 1), semantic chunking was again a higher scoring metric (0.924), but just by a small amount. This is interesting because it shows that other methods do not fall far behind in including semantic ideas within the chunks.

Answer Generation Metrics: While reviewing Table 1, we can see that the *Faithfulness* score which reviews the factual consistency of the generated answer is highest in the paragraph chunking method (0.852). This is interesting because it continues the idea that paragraphs can

be represented as one source of information. By looking at the relevance of the generated answer, paragraph chunking also has the highest scoring (0.800), followed by recursive chunking with (0.722).

We can take a look at the distribution of each metric:

RAGAs Evaluation Metrics Distributions

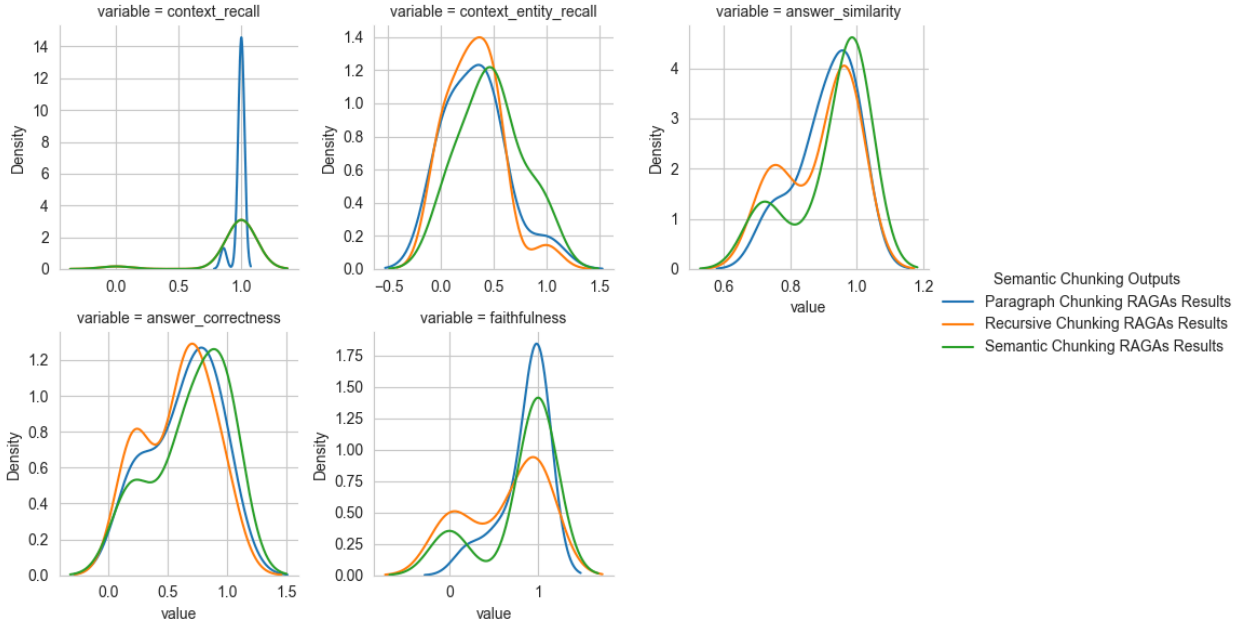


Figure 1(?)**:** Comparison of (MSMarco synthetic data) scores based on their distribution

Overall, depending on the context of how you're implementing RAG, the results show that traditional paragraph chunking is better for providing better generated answers based on ground truth, and semantic chunking is best when reviewing embedding tools. Both are split on implementing chunking methods, with semantic chunking being better able to identify entities.

4.3 Analyst survey chunking task

4.3.1 Chunking Frequency

Across 20 trials (4 documents chunked by all 5 analysts) the amount of chunks made varied from 5-41 (*M*=19.1, *SD*=11.7). Before considering how these chunks varied across documents and participants, we converted these chunks to weighted chunks - accounting for the fact that all four documents ranged in word count. Thus all 20 chunk counts were multiplied by the number of words in that document then divided by the shortest word count (236 words for the intel report). The weighted chunk counts ranged from 4.3-27 (*M*=11.7, *SD*=6.8)

After this transformation we see that there is little variation by document type (Fig. XA). The average weighted chunk counts for each document type ranged from 10.8-12.4 (*SD*=0.7) and we see largely overlapping error bars. In contrast, we see more variation across participants. Each subject described a unique process, and indeed we see average weighted

chunk counts from 5.1-21.6 (*SD*=7.1) (Fig. XB). Additionally the standard error bounds of each participant average are nonoverlapping - while we hesitate to make any statistical significance claims with such a small sample size, this suggests that each participant had a fairly distinct average.

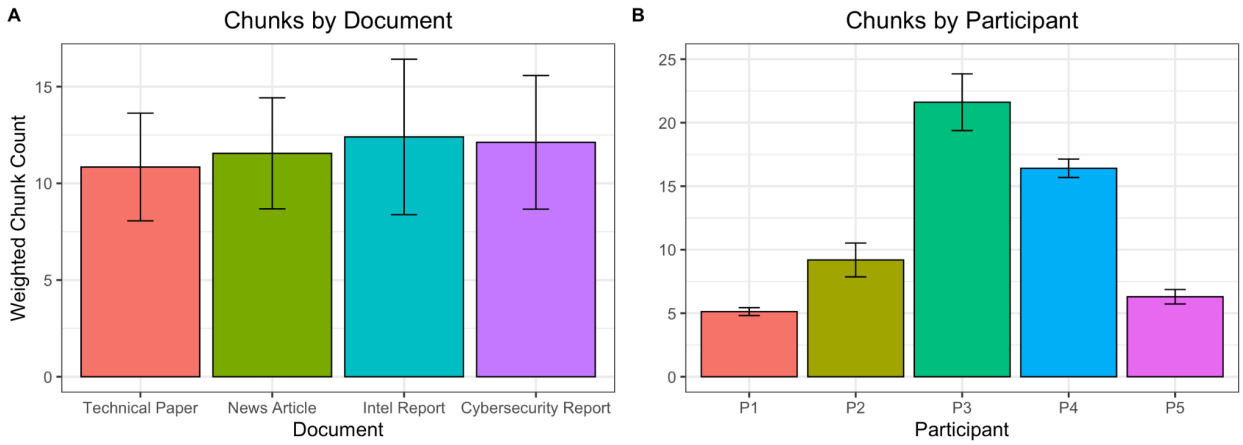


Figure X. *Average chunks made by document and participant weighted for word count.* Each bar represents the average number of chunks made for each document (A) and by each participant (B). Error bars represent standard error. Each participant rated each document, thus for each document N=5 and for each participant there are 4 trials. Chunk counts were standardized by word count as each document varied in length.

4.3.2 Chunking Content

As each analyst had a unique approach to chunking, we first describe each participant's process. The names assigned to them (P1-5) are arbitrary but match their name in Figure XB. P1 primarily made small paragraph sized chunks but sometimes isolated headings or sentences that served as introductions or conclusions. They were fairly consistent in how many chunks they made per document (their weighted chunk count only ranged 4-6). They did, however, make the feast chunks in the two documents they were most familiar with (the news article and the intel report).

For P2, in the first two documents (the technical paper and news article) they would chunk out small details - primarily these appeared to be important details or subheadings and other shifting key phrases (e.g. "Different from previous work"). While many subjects would create chunk boundaries around shifting key phrases, nobody else made the phrase its own chunk, it just signaled the start or end of a chunk for others. In contrast to their pattern for the first two, in the later two documents (the intel and cybersecurity reports) P2 tended more towards small paragraph/sentence sized chunks. This split did not reflect their familiarity as they were most familiar with the second and third documents (the news article and the intel report). One self reported comment they made was that they tend to think hierarchically as they process a document and thus this task was sometimes difficult for them as there was no way to mark up that thought process.

P3's chunking pattern fluctuated within a single document. They'd alternate between leaving a sentence or two intact and chunking up a sentence, indicating there were key details

they wanted to highlight. Interestingly, they would often chunk out function words at the start of a sentence - mimicking the removal of stop words before text is embedded into vectors. They were most familiar with the news article and the intel report but that did not predict chunking frequency.One example of both how they fluctuate within document and how they removed function words is presented here:

Sentence 1: “These”
“Demands,”
“while justifiable from a human rights perspective,”
“pose significant challenges to the peace talks.”

Sentence 2: “The”
“demand for the cessation of human rights violations and the release of political prisoners is a significant sticking point.”

Above they parse sentence 1 into three key points (and one function word likely marked for ignoring). However in sentence 2, which comes later in the same document, they only chunk out the beginning function word. We hypothesize that while both sentences contain about equal amounts of information, the second restates a lot of what is said in the first, and thus it may be less important to break out each piece.

P4 had a very straightforward approach: “find the Essential Elements of Information (EEIs) and chunk those out, ignoring the filler”. EEIs is a common term used in intelligence work referring simply to the 5 Ws - what, when, where, who, why. Analysts often triage documents by looking for those crucial EEIs. P4 was comfortable chunking mid-sentence, often removing the start of sentences. The smallest chunks they made could typically be conceptualized as terms, motives, actions, situations, and potential outcomes - or EEIs. As with others, document familiarity did not predict chunking frequency. They were most familiar with the intel document where they made the least chunks and least familiar with the news article where they made the second least chunks.

P5 typically broke documents up into small paragraphs or single sentences. These single sentences can mostly be conceptualized as either definitions of what the situation is or ongoing threats/predictions of what the situation will become. Document familiarity did not predict chunking frequency, partially because they were fairly familiar with all documents.

Across all five participants there were few consistent patterns. What we did see was that everyone chunked out the headings in the technical excerpt. For all but P1, document familiarity did not appear to drive chunking frequency. Interestingly, a few analysts would chunk out function words at the start of a sentence - mimicking the typical first step when text is embedded into vectors anyways.

Our primary takeaway from these analyst behaviors is that each analyst had different ways of processing documents and thus we think it would be worth generating summaries that are tailored to each analyst's style. For example, P2 might appreciate hierarchical summaries while P4 may prefer a bullet list of EEIs in the document.

5. Discussion

For the summarization task, we expected to see better results from the semantic chunking method due to its attention to semantic groupings. Given the common usage of overlapping sentences, we also did not anticipate it to perform significantly worse than other methods. This raises the need for further exploration into this method. However, due to the fact that `occams` itself does not require embeddings for summarization, we think any differences might be made more obvious by changing to a system that does require embeddings. This is also the benefit of investigating further with RAG and retrieval systems. Despite our limitations in quantitative measures, initial reviews of the summarization UI have been positive. We hope to be able to collect and measure feedback for the summarization UI in the future as a means of collecting qualitative feedback, as well as helping to familiarize users outside of the development team and data science community with the ways the work of data scientists affects the workflows we hope to design for them.

When applied to the RAG task, paragraph and semantic chunk sizes led to mostly similar performance, while we saw worse performance from the recursive method. In our exploration, we found the recursive method to be a common recommendation. Our work brings into question its widespread nature. For semantic chunk sizing, we'd hope to see a greater overall improvement over paragraph given the cost (both computational and literal) required to implement it. However, we give credit to semantic chunk sizing for its higher ability to pull back relevant entities, a metric we valued highly in our process. This work also had its limitations as we had to work with synthetic data, but we would like to see the results with the original MS Marco QA dataset before making further claims in the hopes that we would see greater distinctions.

For the analyst survey, we gained the most insight by interpreting the chunks analysts made in the context of what they reported their process was and how familiar they reported being with the documents. An interesting discrepancy became apparent to us - this task was unlike the way they usually process documents. Often they read looking for EEIs, thus their internal process is more akin to highlighting key details in a document rather than chunking the entirety of a document. While the task in this survey may have been less like what the analysts are used to doing with real documents, we truly intended for them to have to sort the entire document, thus we could more directly compare the output to creating text embeddings for summarizing. Acknowledging this difference allows us to see that for some analysts, chunks are not an indicator that there is one key detail present, but rather a chunk is either something to keep or something to ignore.

6. Future Work

Future work would ideally aim to integrate the two components of work conducted here to let them inform one another. For example, taking patterns we saw in analyst behavior (i.e., breaking up sentences rich with new details into multiple chunks) and letting that serve as a chunking pattern that can be implemented into the UI. Particularly, we would be interested in trying to create summaries based on an analysts chunking behavior, and then actually give

them that tailored summary to see if our analysis was correct and tailoring summaries really would be beneficial.

We would also like to explore how these chunking methods, particularly multilevel, might inform a system that allows analysts to obtain summaries at varying levels of detail for the same document. In our conversations with analysts, we often heard how some analysts prefer a hierarchical exploration of documents, starting with the high-level topic of what the document is about and digging deeper into certain sections as they learn more. A scalable summarization tool may benefit from hierarchical or multilevel chunk size methods in the background.

There are several options that did not make it into our UI, summarization evaluation, or RAG evaluation. We would not only like to see more datasets and summarizers as mentioned, but also more embedding models to see if those have an effect on these tasks using the different chunking methods. We'd also like to try other metrics to add to the robustness of our evaluation, including ROUGE-2 and AutoACU for summarization. For RAG, we would like to consider more deeply how measuring recalled entities could be used to evaluate the effectiveness of retriever systems where the documents do not remain the same size, such as in the chunking case. Given the goal of RAG and optimal chunk sizing to reduce hallucinations, use of hallucination metrics, such as HELM, may be helpful for further evaluation.

A number of future applications of the chunk size question are available for further study, including the implications of languages other than English, both in monolingual and multilingual settings, as well as modes other than text, such as image and audio. These are also areas where further work with analysts to determine how information is broken down may influence how the research proceeds.

References

Ding et al., 2023:

@inproceedings{ding2023revisit,

title={reVISit: Supporting Scalable Evaluation of Interactive Visualizations},

author={Ding, Yiren and Wilburn, Jack and Shrestha, Hilson and Ndlovu, Akim and Gadhave, Kiran and Nobre, Carolina and Lex, Alexander and Harrison, Lane},

booktitle={2023 IEEE Visualization and Visual Analytics (VIS)},

pages={31--35},

year={2023},

organization={IEEE}}

@inproceedings{RAGAs2023, author = {Jithin James, Shahul ES}, title = {RAGAs}, year = {2023}, publisher = {GitHub}, journal = {GitHub repository}, howpublished = {\url{https://github.com/explodinggradients/ragas}}, commit = {4f57d6a0e4c030202a07a60bc1bb1ed1544bf679} }

@inproceedings{RAGAsMetrics, author = {Jithin James, Shahul ES}, title = {RAGAs}, year = {2023}, publisher = {GitHub}, journal = {GitHub repository}, howpublished = {\url{https://docs.ragas.io/en/latest/concepts/metrics/index.html}} }

@inproceedings{langchain2022, author = {Chase, Harrisn}, title = {Langchain}, year = {2022}, publisher = {GitHub}, journal = {GitHub repository}, howpublished = {\url{https://python.langchain.com/v0.1/docs/modules/data_connection/document_transformers/} }

@inproceedings{RAGEfficiency, author = {Sthanikam Santhosh}, title = {Enhancing RAG Efficiency: The Power of Semantic Chunking}, year = {2024}, publisher = {Medium}, journal = {Towards Data Science}, howpublished ={\url{https://medium.com/@sthanikamsanthosh1994/enhancing-rag-efficiency-the-power-of-semantic-chunking-844f9cfbdd0b} }

@inproceedings{semanticRAG, author = {Plaban Nayak}, title = {Semantic Chunking for RAG}, year = {2024}, publisher = {Medium}, journal = {Towards Data Science}, howpublished ={\url{https://medium.com/the-ai-forum/semantic-chunking-for-rag-f4733025d5f5} }

@inproceedings{wu-etal-2020-mind,

title = "{MIND}: A Large-scale Dataset for News Recommendation",
author = "Wu, Fangzhao and
Qiao, Ying and
Chen, Jiun-Hung and
Wu, Chuhan and
Qi, Tao and
Lian, Jianxun and
Liu, Danyang and
Xie, Xing and
Gao, Jianfeng and
Wu, Winnie and
Zhou, Ming",
editor = "Jurafsky, Dan and
Chai, Joyce and
Schluter, Natalie and
Tetreault, Joel",
booktitle = "Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics",
month = jul,
year = "2020",
address = "Online",
publisher = "Association for Computational Linguistics",
url = "https://aclanthology.org/2020.acl-main.331",
doi = "10.18653/v1/2020.acl-main.331",
pages = "3597--3606"}