



**School of Computer Science**

FACULTY OF ENGINEERING AND PHYSICAL SCIENCES

**UNIVERSITY OF LEEDS**

---

## **Final Report**

### **Generating Structured Datasets via Systematic LLM Prompting: A Framework for Synthesising Text Data to Train NLP Models Used in Analysing Customer Feedback**

**Oliver Alexander Hague**

Submitted in accordance with the requirements for the degree of  
**BSc. Computer Science with Artificial Intelligence**

**2024/2025**

**COMP3931 Individual Project**

The candidate confirms that the following have been submitted:

<b>Items</b>	<b>Format</b>	<b>Recipient(s) and Date</b>
<i>Final report</i>	<i>PDF file</i>	<i>Uploaded to Minerva (30/04/25)</i>
<i>Electronically signed consent forms</i>	<i>File archive</i>	<i>Uploaded to Minerva (30/04/25)</i>
<i>Link to online code repository</i>	<i>URL</i>	<i>Sent to supervisor/assessor (30/04/25)</i>

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student)  \_\_\_\_\_

## Summary

The framework developed in this project is designed to accept a highly structured set of user requirements and systematically generate a collection of individual text units using a large language model, which can be used as precisely defined training data. This is driven by the rising demand for more comprehensive textual datasets that are needed to train specialised language analysis tools. These are particularly effective in the e-commerce sector to help automate the evaluation of customer feedback, which serves as the context for this project.

The final pipeline primarily focuses on allowing users to define the exact topic and sentiment distribution across the final body of text. For example, one could define the need for 10% of the final text corpus to describe customer service expressed in a negative tone and so on until the full contents have been outlined. This is achieved by dividing the problem down into smaller subtasks, generating the text for each unit and combining the results to create the final corpus. The main principle of this approach is to break down the generation process into smaller tasks, thereby enhancing the accuracy and alignment with the initial requirements.

This functionality is housed within a user-friendly web application and managed with a simple file handling system to organise requirement sets and any synthetically generated text. Each organisation utilising this tool is expected to access their own large language model through an integrated API function and define their own prompting strategy. This helps to provide a level of flexibility and suit individual contextual needs. Nonetheless, the core algorithm that handles the translation from an abstract set of requirements to a strictly defined group of textual units is robust and efficient, providing a solid foundation for the generation process.

## **Acknowledgements**

I would like to primarily thank my project supervisor Dr. John Stell for his continued support throughout the development process. His perspectives provided me with unique insights and helped to shape the project's trajectory.

I am also grateful to my assessor, Dr. Sebastian Ordyniak, whose valuable advice during our meeting helped me to refine the most programmatically challenging aspects of the project.

Lastly, I would also like to express my gratitude to my close family and friends for supporting me in my personal life and constantly driving me to do my best.

## Table of Contents

<b>Summary.....</b>	<b>iii</b>
<b>Acknowledgements.....</b>	<b>iv</b>
<b>Table of Contents .....</b>	<b>v</b>
<b>Chapter 1 – Introduction and Background Research .....</b>	<b>1</b>
1.1 Project Focus.....	1
1.1.1 Core Proposal .....	1
1.1.2 Rationale and Motivation .....	1
1.2 Underlying Context .....	1
1.2.1 Defining NLP .....	1
1.2.2 Customer Feedback Analysis .....	2
1.2.3 Dataset Quality.....	3
1.3 Literature Review .....	5
1.3.1 LLM Output Supervision .....	5
1.3.2 Data-Driven Synthesis.....	6
1.3.3 Dataset Creation Pipelines .....	6
1.4. Critical Analysis .....	7
1.4.1 Prevailing Limitations .....	7
1.4.2 Aims and Objectives.....	7
<b>Chapter 2 – Methodology.....</b>	<b>8</b>
2.1 Proposed Solution .....	8
2.1.1 Conceptual Framework .....	8
2.1.2 System Architecture .....	9
2.1.3 Implementation Philosophy .....	9
2.2 Algorithmic Design .....	10
2.2.1 Rulebook Interface .....	10
2.2.2 Chunk Partitioning .....	12
2.2.3 Collection Forming .....	12
2.2.4 Text Generation .....	14
2.2.5 User Interface.....	15
2.3 Project Management.....	15
2.3.1 Requirements List .....	15
2.3.2 Agile Workflow .....	16
2.3.3 Version Control .....	16

<b>Chapter 3 – Implementation and Validation .....</b>	<b>17</b>
3.1 Technical Configuration .....	17
3.1.1 Programming Languages .....	17
3.1.2 Imported Libraries .....	17
3.2 Algorithmic Implementation.....	17
3.2.1 Rulebook Interface .....	17
3.2.2 Chunk Partitioning .....	18
3.2.3 Collection Forming .....	19
3.2.4 Text Generation .....	22
3.2.5 User Interface.....	23
<b>Chapter 4 – Results, Evaluation and Discussion .....</b>	<b>25</b>
4.1 Evaluation.....	25
4.1.1 Output Text Quality .....	25
4.1.2 Performance Metrics .....	25
4.1.3 Usability and Integrity .....	27
4.2 Conclusion.....	28
4.3 Future Work.....	29
4.3.1 Output Validation.....	30
4.3.2 Alternative Applications .....	30
<b>List of References .....</b>	<b>31</b>
<b>Appendix A – Self-appraisal .....</b>	<b>36</b>
Section 1 – Self-Appraisal.....	36
1.1 Development Process .....	36
1.2 Personal Reflection .....	36
1.3 Key Takeaway .....	37
Section 2 – Potential Concerns.....	38
2.1 Legal Issues.....	38
2.2 Social Issues.....	38
2.3 Ethical Issues.....	39
2.4 Professional Issues .....	39
<b>Appendix B – External Materials .....</b>	<b>40</b>
<b>Appendix C – Supporting Materials .....</b>	<b>41</b>
Section 1 – Methodology .....	41
1.1 Synthetic Dataset Generation Pipeline Diagram.....	41
Section 2 – Implementation .....	42

2.1 Unit Testing and Code Explanation .....	42
2.2 User Interface and Experience .....	46
Section 3 – Evaluation .....	54
3.1 Generated Text Examples.....	54
3.2 Simulated Annealing – Configuration Comparison .....	56
3.3 Simulated Annealing – Sprint 2 vs. 4 Implementation.....	56
3.4 Participant Information Sheet .....	57
3.5 Participant Task Sheet .....	59
3.6 Participant Consent Form.....	61
3.7 Participant Feedback Form .....	62

## **Chapter 1 – Introduction and Background Research**

### **1.1 Project Focus**

#### **1.1.1 Core Proposal**

This project aims to establish a structured framework for generating synthetic datasets using large language models (LLMs), specifically for creating synthetic customer reviews to aid in the training of natural language processing (NLP) models utilised in the commerce sector. A key feature involves accepting user-defined requirements that specify the exact profile of the desired dataset, including parameters such as size, topical focus, and sentiment distribution, which are then automatically translated into individual dataset element specifications.

Based on this structure, a series of precisely formulated prompts are programmatically rendered and processed by an LLM to produce the complete dataset. The goal of this structured prompting, guided by the detailed outline, is to achieve a superior degree of correspondence between the final generated text corpus and the initial user specifications.

#### **1.1.2 Rationale and Motivation**

The primary motivation revolves around addressing the increasing data demands for training sophisticated NLP models (Ghaisas & Singhal, 2024). Existing datasets frequently reveal limitations in terms of their size and contextual alignment with the distinctive requirements of individual commercial applications. This often necessitates the creation of tailored datasets adjusted to organisational needs for effective model training and evaluation (Lu et al., 2023).

Therefore, the proposed framework seeks to enhance the efficiency and scalability of dataset creation. Most traditional methods that involve manual data collection and annotation are uneconomical and can become bottlenecks in the development cycle (Amrish & Shwetank, 2024). By establishing the foundation for an automated synthetic data generation pipeline, this project aims to enhance the efficiency of this process. The goal is to build an adaptable framework to fulfil the increasing needs for training new customer feedback NLP models.

### **1.2 Underlying Context**

#### **1.2.1 Defining NLP**

Natural language processing, a branch of artificial intelligence (AI), is the study of complex algorithms and models that enable computers to recognise, understand, and generate text in human languages (Stryker & Holdsworth, 2024). There are several different types of NLP tools tailored to different applications, such as text classification (e.g. sentiment analysis), entity recognition (e.g. feature extraction), and translation. This technology can automate numerous manual processes, such as customer feedback analysis, the focus of this project.



Unlike traditional deterministic programs that follow predefined rules, NLP technology works by attempting to learn the relationships between different words and sentences from vast amounts of textual data (Stryker & Holdsworth, 2024). This process, often referred to as "training", embeds patterns observed from the provided examples within statistical models, thereby allowing them to make predictions for unseen data using their acquired knowledge.

### 1.2.2 Customer Feedback Analysis

The evaluation of customer feedback provides significant business value by identifying crucial insights into consumer preferences, pain points, and the overall brand perception (Sokolovsky, 2023). This business intelligence directly informs product development and can shape impending marketing strategies. Furthermore, customer reviews also significantly impact sales as consumers increasingly use them to aid with purchase decisions (Park et al., 2019). Consequently, analysing written reviews is vital for modern businesses to succeed.

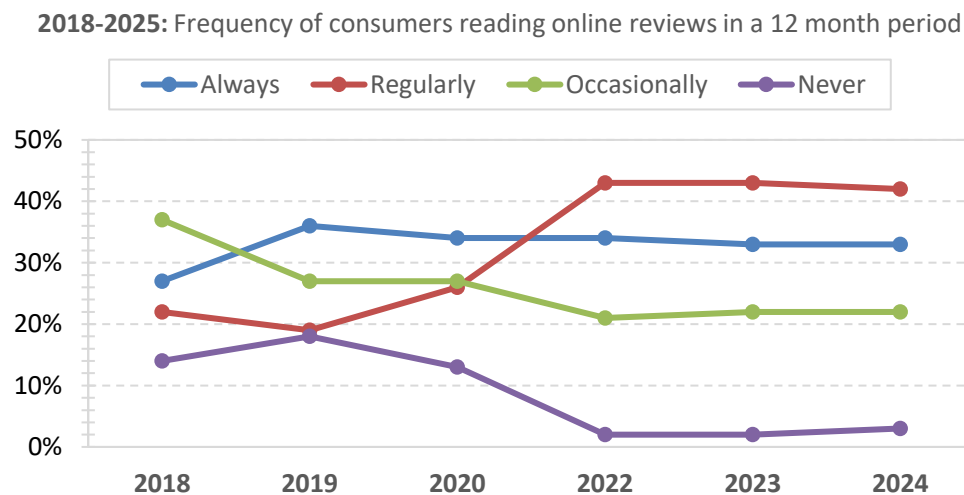


Fig. 1.1 – Increasing customer engagement with product reviews (Paget, 2025)

#### 1.2.2.1 Prior approaches

Initial efforts to automate this process utilised emerging NLP techniques prevalent at the beginning of the 21st century. Notably, in 2004, researchers at the University of Illinois (Hu & Liu, 2004) introduced an innovative methodology for extracting insights from online reviews. Their approach employed early forms of entity recognition (identified as part-of-speech tagging) and sentiment analysis (leveraging WordNet – a large lexical database) to focus on product features and their related sentiment, moving beyond general review summarisation.

Building upon this foundation, subsequent research from Simon Fraser University (Wang & Ester, 2014) developed models capable of predicting numerical ratings for unscored reviews. Despite these advancements in the field of NLP, the main analytical techniques of this era lacked the capacity for more comprehensive contextual understanding. This resulted in limited accuracy and a lack of distinct and innovative progress (Wang & Ester, 2014).

### **1.2.2.2 Advent of Transformers**

A research paper released in 2017, titled "Attention is All You Need" (Vaswani et al., 2017), prompted a paradigm shift in NLP, moving the field away from the established recurrent neural networks (RNNs) towards a new transformer architecture. RNNs process text serially and consequently often struggle with long-range contextual understanding. In comparison, transformers implement an attention mechanism that allows them to directly access and weigh the significance of any word within the input, providing a broader scope of context.

The introduction of transformer-based architectures has directly contributed to the increase in academic research focused on applying this technology to improve upon current techniques for analysing customer feedback (Malik & Bilal, 2024). This surge highlights the potential for transformers to enhance current practices, such as opinion mining and sentiment analysis.

### **1.2.2.3 Present Landscape**

Leveraging the transformer architecture, LLMs stand out as notably impactful tools thanks to their remarkable power and flexibility. These versatile models are trained on immensely large datasets and excel at tasks such as engaging in conversation, summarising information and translating across different languages, while only requiring simple prompting (IBM, 2023). Given this level of performance, they are often used for various feedback analysis tasks over developing narrowly scoped NLP tools (Korkankar et al., 2024; Siledar et al., 2024).

Nonetheless, a significant limitation of the multi-purpose architecture with which LLMs are built is the substantial demand on resources and often require specialised servers to run even a single instance (Naveed et al., 2023; Patibandla, 2024). Despite broad accessibility thanks to cloud computing, using highly capable LLM models, such as OpenAI's GPT-4, can quickly incur high costs, particularly when handling large quantities of data (Benram, 2024).

Moreover, general purpose LLMs present challenges regarding inherent biases, as different models tasked with the same job often perform inconsistently (Guo et al., 2024). Addressing these limitations requires fine-tuning these models to better suit contextual needs. This is the case with both LLMs and specialised NLP tools (Bronsdon, 2024; Rojo-Echeburúa, 2024).

While the advantages of modern NLP for enhancing business efficiency are clear (Tian et al., 2024), fine-tuning a model's behaviour hinges on the availability of datasets used to adapt its source of knowledge. This emphasises the critical and ongoing need for new training data.

## **1.2.3 Dataset Quality**

The quality of the training dataset directly impacts the model's performance. It serves as the foundation upon which models learn new patterns and relationships relevant to the task they are needed to perform. Thus, low-grade data can lead models to struggle with unseen inputs.

### **1.2.3.1 Open-Source Datasets**

A number of open-source corpora containing labelled examples of customer reviews are publicly available and provide a useful foundation for many NLP projects. Commonly used is the Amazon reviews dataset (McAuley, 2023), which contains over 500 million items from the years 1996-2023. Other noteworthy examples include the Yelp Open dataset (Yelp, 2025) and the IMDB ratings dataset (IMDB, 2024). These resources are also typically free to use and are often referenced in academic research, helping to provide a standard comparison.

However, there are several drawbacks when considering their application in a commercial context. Primarily, licences for most of these datasets largely restrict usage to educational purposes. Moreover, these corpora are still broad and generally lack a detailed focus. For instance, attempting to train a model to identify technical specifications in laptop reviews by using a dataset for movie ratings would be ineffective and thus better options are required.

### **1.2.3.2 Current Inadequacies**

Lack of a domain-centric focus is a common issue hindering the performance of NLP in task-specific settings (Kim Amplayo et al., 2022). Furthermore, current datasets can also be limited by factors such as mislabelled data, similar or repeated examples, and outliers (Mehta, 2024). Regarding customer review analysis, while robust sentiment classification and topic modelling datasets are available, developing more powerful NLP models for deeper customer feedback insights requires a new generation of text datasets (Malik & Bilal, 2024).

For example, newer specialised datasets can enable emerging multilingual NLP models to empower e-commerce with analysis tools previously exclusive to English-speaking customer bases (Otten, 2023). However, perhaps more importantly, there is also a vital need for new data to train NLP tools capable of detecting synthetically generated text, as advanced LLM models have exacerbated issues relating to the spread of fake reviews (Poojary, 2024).

### **1.2.3.3 Creating New Datasets**

Manually curated datasets allow for highly tailored domain coverage and exact linguistic styles but are considered costly due to human labour (Graziani et al., 2025). In comparison, synthetic data generation offers a cheaper solution to meet the increasing demand for large and specialised corpora, especially given the recent advancements in LLM technology.

However, synthetic datasets present a set of different challenges, such as overfitting, biases, unnatural language, and improper data classification labels (Hao et al., 2024). Such issues can occur when automated pipelines do not implement adequate control and supervision over the generation process. With these limitations in mind, synthetic data has the potential to enable organisations to develop highly specific custom datasets previously unobtainable, subsequently promoting the development of more accurate and reliable NLP solutions.

## 1.3 Literature Review

Having established the context for customer feedback analysis using NLP and the need for better and more relevant training datasets, it is important to consider how researchers are attempting to mitigate the challenges associated with synthetic data and to show the current lack of solutions that enable the controlled generation of large domain-focused text datasets.

### 1.3.1 LLM Output Supervision

Ensuring LLMs are suitable for specific tasks like dataset creation requires the transition from unstructured text generation towards more systematic approaches. One method focuses on controlling specific textual elements, such as product features or names. Amazon research (Aggarwal et al., 2022) demonstrated a question-and-answer prompting technique to insert particular entities into text, showing promise for improving named entity recognition tasks where real data is scarce. However, while this methodology forces entities to appear in a text corpus, it lacks control over context (e.g. sentiment) and content proportion distributions.

Ensuring factual correctness is another factor to consider when creating synthetic datasets. Although the generation of training data was not its direct aim, academic research focused on different summarisation techniques proposed a way of artificially populating text with known facts, serving as a control for evaluation (Ahn & Khosmood, 2022). Whilst exhibiting a robust method for injecting content into a synthetic text corpus, it requires domain-specific grammar rules, serving as a costly barrier for entry to many organisations.

Other approaches also showcase the strength of direct instruction prompting with augmented LLMs. A recent study illustrated the improved accuracy and compliance of fine-tuned models (Zhou et al., 2023), indicating the feasibility of less intricate pipelines for generating synthetic data that adhere to lexical and stylistic needs. While it shows the potential of LLMs to create highly tuned outputs, it does not offer a direct framework for producing NLP training datasets.

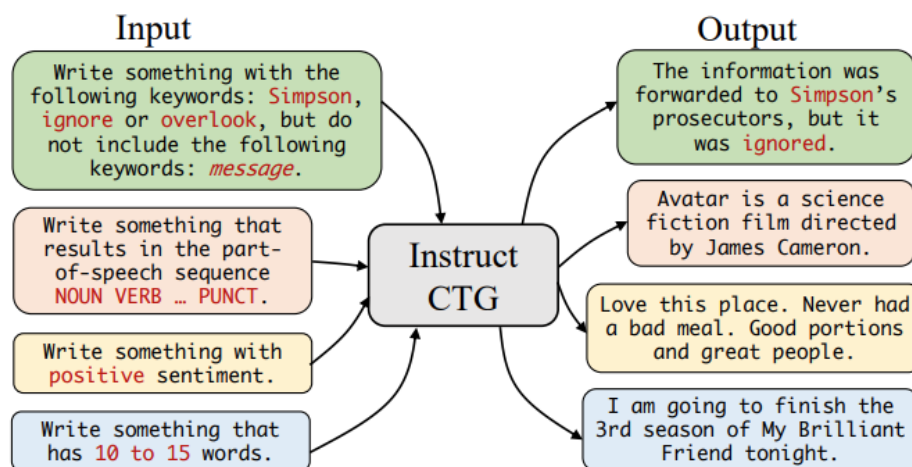


Fig 1.2 – Capabilities of the fine-tuned "Controlled Text Generation" LLM by (Zhou et al., 2023)

### **1.3.2 Data-Driven Synthesis**

Another challenge lies in generating expansive text corpora with structured input data, rather than relying on manual prompt construction, which is time-consuming and limits scalability. This task is crucial to enable rapid assembly and iteration of specially designed datasets, such as realistic customer reviews uniquely tailored to represent different audiences.

Research conducted in the Czech Republic evaluated the performance of several LLMs in generating written text passages based solely on simple instructions and tabular data, such as generating a product description given a set of specifications (Kasner & Dusek, 2024). Although the models delivered coherent text, many contained notable semantic and factual inaccuracies, showing that LLMs struggle to work with prompts containing too much detail. While their process expedites synthetic text generation from structured data, the accuracy issues would hinder its effectiveness for creating and validating thorough NLP training sets.

### **1.3.3 Dataset Creation Pipelines**

A few complete solutions are already available for the generation of customised datasets. The framework developed by (He et al., 2022) uses LLMs to generate unlabelled text for specific applications, which is then classified by other NLP models to automatically tag the synthetic data. A number of such datasets were used to train other models, showing notable improvements on benchmark tasks. Despite these results, the first create and then classify methodology does not allow for complete control over the final dataset and its characteristics. Moreover, the quality of the output is limited by the performance of the NLP classifiers used.

There also exist commercially available tools that harness LLMs to provide complete training datasets. Examples of a few leading platforms include Mostly AI (Aysha, 2023) and Gretel AI (Segbroek et al., 2025), which provide extensive pipelines for generating realistic synthetic data. Nevertheless, a significant constraint is the required dependence on their platform as well as an abstraction layer that limits customisability and clarity in the process. Furthermore, these solutions are also restricted by fees for any form of meaningful commercial utilisation.

Additionally, there are several open-source frameworks available for generating synthetic text. Hugging Face offers a no-code application designed for generating textual datasets with the use of instructional prompts (Berenstein et al., 2024). Despite its accessibility, the framework lacks robust configuration options needed to enforce precise structural constraints across an entire dataset. In contrast, frameworks like LangChain (LangChain, n.d.) provide a set of libraries for building complex LLM pipelines, including tools for dataset generation. Its pre-programmed components for prompt templating, model communication, and output handling allow for the development of robust programs built for creating synthetic datasets. However, LangChain fundamentally is a software toolkit and does not provide a ready-built solution for users to convert abstract requirements into datasets that align with their needs.

## **1.4. Critical Analysis**

### **1.4.1 Prevailing Limitations**

The literature review shows various techniques for managing LLM outputs and constructing generation pipelines. However, there are several limitations hindering the clear-cut creation of large and highly specific synthetic datasets. Studies often focus on controlling individual traits, like ensuring that specific entities appear or following stylistic parameters but fail to provide detailed control over the profile of the entire final text corpus (e.g. topic/sentiment).

Furthermore, when provided with structured numerical requirements, LLMs have still been shown to produce incorrect output when dealing with large quantities of data. This implies the need for a system capable of dividing complex constraints into a set of simplified sub-tasks.

Additionally, there exists a noticeable gap between available software and practical solutions. Open-source frameworks, like LangChain, are powerful and can be easily tailored to meet individual needs but do not currently offer a standalone dataset generation tool. Moreover, other commercialised platforms offer complete solutions but come with added costs and limited customisability. This leaves a need for an accessible and flexible synthetic dataset generation framework that is configurable without requiring extensive coding proficiency.

### **1.4.2 Aims and Objectives**

The primary goal is to bridge the gap between a set of abstract high-level requirements and the detailed orchestration needed for dependable LLM text generation.

1. Developing a sophisticated input system that can accept conceptual parameters, such as the distribution of different topics, sentiment and size of individual elements.
2. Designing the logic for a structuring algorithm that processes the input and translates it into a set of concise generation sub-tasks, each paired with a tailored LLM prompt.
3. Implementing a standardised system that is able to handle the bulk generation of text using an LLM in parallel with efficient resource control and error handling.
4. Providing a basic graphical interface that enables users to run tasks, export the final generated text as well as visually analyse the characteristics of individual datasets.

Achieving these goals will result in a framework that allows non-technical users to completely control the contents of the final text corpus whilst also providing a foundation for further development. In the context of customer review analysis, the application aims to empower organisations to create extremely focused datasets, allowing for the development of a family of fine-tuned NLP models, each tailored for specific contexts to ensure optimal performance.

## Chapter 2 – Methodology

### 2.1 Proposed Solution

#### 2.1.1 Conceptual Framework

The proposed solution addresses the inadequacies identified in existing synthetic data generation pipelines. The following framework accepts structured user requirements while enabling generative flexibility, primarily by allowing the user to define the distribution of topics and sentiments across the final dataset. These constraints are processed by the pipeline to enable precise control over instructions rendered in the final prompts, leveraging the generative capabilities of modern LLMs, which were highlighted in the literature review.

##### 2.1.1.1 Text Management

The methodology structures the output dataset using two fundamental building blocks.

**Chunks** – Discrete non-divisible segments of text defined by three attributes.

- **Topic** – The single matter discussed within this snippet of text. Can be any user-defined string e.g. “*Performance*” or “*Connectivity*” for a set of laptop reviews.
- **Sentiment** – The tone and form of expression for the text snippet. Limited to one of three options specific to this project: “*Positive*”, “*Neutral*”, “*Negative*”.
- **Word Count** – The target word count of the text snippet. This can be any positive integer within a user-defined range for possible chunk sizes.

Simply put, a chunk represents a single reference to an idea in a given sentiment bounded by a word count. For example, assuming 100 chunks with topic *A* and sentiment *X* are created, there will be 100 unique references to that topic-sentiment pair in the final dataset.

**Collections** – A High-level grouping of chunks that represent complete text units, such as an individual customer review in the case of this project.

An additional hard constraint enforcing topic uniqueness within each collection helps to prevent contradictory sentiment expressions about the same topic. This helps to ensure logical coherence within each collection (i.e. dataset element) in the final synthetic corpus.

##### 2.1.1.2 Key Strengths

This method of decomposing the structure of the proposed dataset provides several benefits. By dividing the complex generation task into smaller and more strictly defined subtasks, it allows for strict control over the synthetic output. This helps to significantly limit the hallucination problem common with unconstrained and broad LLM tasks (Xu et al., 2024).

The chunk-based methodology also aligns with how humans naturally compose product reviews as a list of product aspects together with their opinions on each one (Keshavarzi et al., 2024). Furthermore, the divide and conquer approach also allows for systematic validation throughout the generation pipeline to ensure consistency and correctness.

Moreover, the framework employs generalised terminology (“*chunks*” and “*collections*”) to ensure adaptability across different domains beyond customer reviews. This allows it to be used in other settings without requiring major adjustments to the software or documentation.

### 2.1.2 System Architecture

The solution implements a modularised architecture to separate key logic. Clear boundaries between system components allow for independent optimisation or replacement without requiring major code restructuring. A separate user interface will enable easy and safe access to functionality within each layer. Each component is outlined in the figure below.

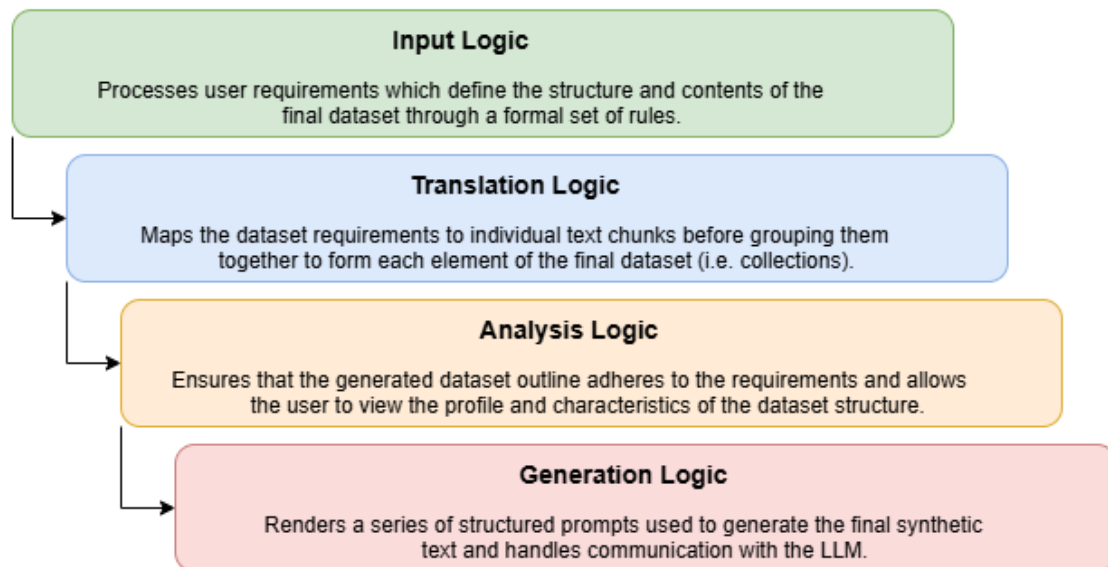


Fig 2.1 – Core components within the proposed solution (Created with Draw.io)

### 2.1.3 Implementation Philosophy

The framework is built to leverage LLMs through precise prompting and orchestration to enhance its generational capabilities rather than to compete with them. To ensure longevity, it is crucial to build synthetic data generation tools that advance alongside LLMs, especially considering the rate of improvement recently seen in this space (Nadas et al., 2025).

However, the solution does not attempt to fine-tune, filter or verify the content generated by LLMs for the final corpus. The factual accuracy of the text and validity of any domain-specific knowledge relies on the capacity of the selected LLM and is outside the scope of this project.



While the addition of such functionality to the framework would greatly enhance the quality of the synthetic data generated by the LLMs, completing the project within the allocated timeframe would not be feasible. Although the pipeline will be capable of generating text using the constructed prompts, the final semantic alignment or validity will not be evaluated.

This platform serves as a middle ground between manual and fully automated synthetic dataset creation. It is built to be flexible and iterated upon based on the needs of individual organisations, such as the integration of custom LLM models to suit different demands.

## 2.2 Algorithmic Design

### 2.2.1 Rulebook Interface

Firstly, the user must formalise their requirements, outlining the overall profile and content distribution they aim to achieve in the final output. In an informal sense, the user should be able to state preferences like 10% of the dataset should focus on topic  $A$  in sentiment  $X$ , 5% for topic  $A$  in sentiment  $Y$ , 15% for topic  $B$  in sentiment  $X$  etc. until fully outlined by the user.

This information about user requirements will be captured within an object referred to as a "*rulebook*." The rulebook functions as the central blueprint and enables structured definition of datasets, which can scale to a variety of use cases beyond customer review datasets.

The user first determines how the size of the corpus is measured, depending on the mode:

- **Word Mode** – The final corpus aims to sum up to a specific word count.
- **Chunk Mode** – The final corpus is comprised of a total number of chunks.

Following this, the user will fill out two tables, which define the characteristics of the dataset.

#### 2.2.1.1 Content Constraints

The first table is for defining the topic-sentiment distribution, which describes the profile of the synthetic customer reviews. Each row corresponds to a topic, while columns capture properties and constraints regarding the specified topic and its associated chunks.

- **Topic Name** – A descriptive label or short string detailing a given aspect of the product/service, which will be rendered in individual prompts for the LLM to process.
- **Proportion** – A real number  $p_i \in [0,1]$ , indicating how much of the entire corpus (words or chunks) belongs to the given topic. Where  $\sum_{i=1}^T p_i = 1$  always, given that  $T$  is the total number of topics and  $p_i$  is the proportion of the  $i$ -th topic.
- **Sentiment Ratios** – For each topic, its allocated share of the dataset total is further partitioned by *negative*, *neutral*, and *positive* sentiments. This is represented as a three-tuple  $(r_1, r_2, r_3)$ , where  $r_1, r_2, r_3 \in [0,1]$  and  $r_1 + r_2 + r_3 = 1$  always.

- **Word Count** – Specifies a permitted  $\min_i \in \mathbb{Z}^+$  and  $\max_i \in \mathbb{Z}^+$  word count that each chunk of given topic must abide by, where  $\min_i < \max_i$  for any  $i$ -th topic.
- **Chunk Characteristics** – Additional settings that allow the user to specify different size preferences for chunks within the same topic (detailed in the next chapter).

These constraints are later used to partition the required size of the dataset into chunks.

The following diagram illustrates an example of content distribution across different topics and sentiments, given a total word count. The leftmost bar represents the total dataset size, while the rightmost bars show the word count assigned to each topic-sentiment pair.

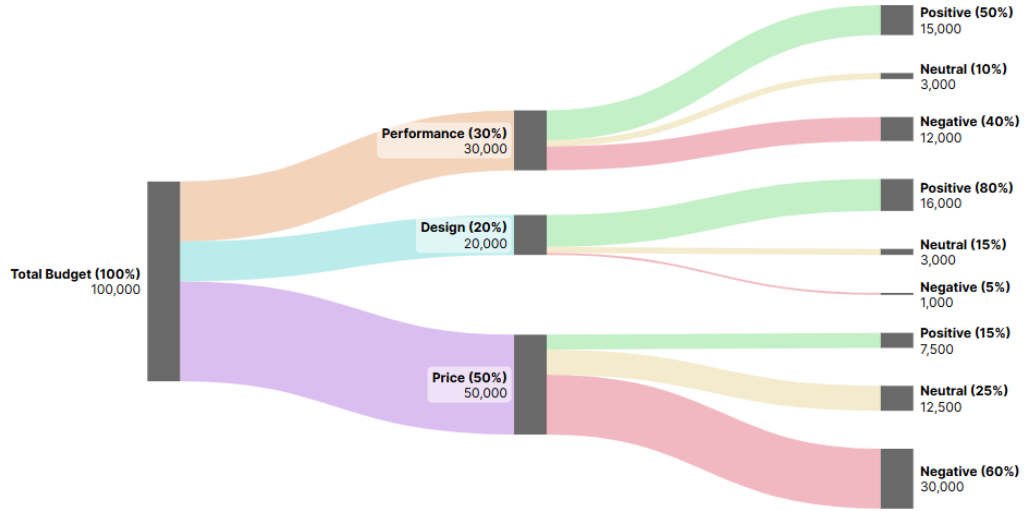


Fig. 2.2 – Topic-sentiment word count distribution example (Created using sankeyart.com)

### 2.2.1.2 Collection Constraints

After partitioning the text into chunks, they need to be aggregated into collections to form individual dataset elements. To maximise the feasible number of solutions under the topic uniqueness constraint, user parameters should ideally avoid setting impossible requirements. User control over collection size allows for this, as it significantly influences the dataset without heavily limiting chunk combinations (as would be the case with topic conflict classes).

Furthermore, research indicates that element size in training data significantly impacts NLP performance (Mehrafarin et al., 2022). Since the primary purpose of this pipeline is synthetic data for model training, user control over final collection size is a vital and relevant feature.

Therefore, the rulebook's second table allows users to define size ranges for collections.

- **Interval Table** – Specifies a set of  $N$  size ranges  $[S_k, E_k] \subseteq \mathbb{Z}^+$  (either word count or chunks) where  $S_k \leq E_k$  and  $S_{k+1} = E_k + 1$ . Additionally, a real number  $q_k$  indicates the fraction of collections that should fall within that range, where  $\sum_{k=1}^N q_k = 1$  always.

### 2.2.1.3 Rationale and Advantages

By structuring requirements around topic-sentiment proportions and global collection size bounds, the rulebook provides both precise control as well as high-level organisation over completed units. Moreover, it serves as a source of truth to compare the output with.

Furthermore, simple validation via proportion and ratio summation allows for easy validation of the rulebook's contents. Bearing these factors in mind, users can incorporate diverse topics and text size restrictions to create a corpus reflecting the nuances of real-world data.

Additionally, the rulebook's general structure and easily expandable tables allow for its application beyond synthetic review generation whilst maintaining a consistent methodology.

## 2.2.2 Chunk Partitioning

Chunk partitioning addresses the fundamental task of dividing each topic-sentiment size allocation into a set of chunks that follow the user's preference requirements.

### 2.2.2.1 Quantity of Chunks

Given a user rulebook with a fixed word budget  $W$  for an  $i$ -th topic-sentiment combination, the system must determine the necessary number of chunks to divide the budget into.

The minimum feasible chunks  $\lceil W/\max_i \rceil$  and maximum feasible chunks  $\lfloor W/\min_i \rfloor$  establish mathematical boundaries, which determine whether there exists a valid partitioning of chunks given the allocated word count and size restrictions. Assuming there exists a valid partition, a user preference parameter then determines the final number from the options available.

Alternatively, this step is skipped when the number of chunks is defined using chunk mode.

### 2.2.2.2 Allocating Word Counts

Given a chosen number of chunks, the partitioning algorithm ensures constraint satisfaction by initially assigning minimum word counts to all chunks. Subsequently, the remaining word budget for the topic-sentiment pair is distributed across all the elements. This distribution considers a separate user preference parameter for determining size variance.

## 2.2.3 Collection Forming

Grouping chunks is a complex combinatorial problem. Without any constraints,  $n$  distinct chunks yield  $2^n$  potential combinations, resulting in exponential growth. Even with added constraints limiting possible solutions, it is still a challenging problem. Formally, given:

- Set of chunks  $C = c_1, \dots, c_n$ , each with an assigned topic  $t(c_i)$  and word count  $w(c_i)$ .
- Soft constraint defining size ranges and their target proportions in the final dataset.

The goal is to partition  $C$  into sets of chunks  $P = \{P_1, \dots, P_m\}$  such that the following is met:

1. Each chunk is accounted for and assigned to exactly one collection in the solution.

$$\cup P_i = C \text{ and } P_i \cap P_j = \emptyset \text{ for any } i \neq j.$$

2. No set contains multiple chunks with the same topic to respect the hard constraint.

$$\forall c_a, c_e \in P_i, t(c_a) \neq t(c_e).$$

3. Minimise the difference between actual  $q_k$  and target  $t_k$  fractions, given N ranges.

$$\sum_{k=1}^N |q_k - t_k| \approx 0.$$

### 2.2.3.1 Computational Complexity

The collection forming problem exhibits characteristics of several well-known NP-hard problems. It can be compared to the Variable Sized Bin Packing Problem (VSBPP), a generalisation of the classical bin packing problem known to be NP-Hard (Blum et al., 2010).

The formal definition of VSBPP is as follows:

- Given a set  $S$  of  $n$  items, where each item  $i \in S$  has a weight  $w_i > 0$ .
- Given a set  $B$  of  $m$  different bin types, where each bin type  $k \in B$  has a capacity  $W_k > 0$  and a cost  $c_k$ .
- The goal is to minimise  $\sum (k \in B) y_k \cdot c_k$  where  $y_k$  is the number of bins of type  $k$  used, subject to packing all items from  $S$  into a finite number of bins.

The collection forming problem can be reduced to VSBPP by assuming the following:

- Each chunk is an item in  $S$ , where the weight is either  $w(c_i)$  or 1 (when the rulebook measures the number of chunks instead of word count). For simplicity, we assume that every chunk has a unique topic within  $S$ .
- Each size interval represents a bin type  $k$  and has a dynamic cost based on its deviation from its target usage. For example, given a target fraction  $t_k$  of the total number of bins and a real fraction  $q_k$ , then  $c_k = |q_k - t_k| \cdot y_k$  for the current solution.
- Similar to VSBPP, the goal is to pack chunks into collections of different sizes (either total word count or chunk count) to minimise the total cost of the solution.

This reduction demonstrates that the collection forming problem is comparable in complexity to VSBPP, potentially suggesting its NP-hard nature. Although, the explanation above cannot be considered as a formal mathematical proof, it underscores the difficulty of the problem.

### 2.2.3.2 Approximation Approach

For this reason, exploring all valid partitions in polynomial time is infeasible for large inputs and efficiently searching for optimised solutions bounded by some limit is the best approach.

Greedy algorithms, although extremely fast, can easily entrap themselves by making poor decisions at early stages (Johnson et al., 1974). This would often provide suboptimal solutions, defeating the framework's purpose of custom size ranges. In contrast, simulated annealing (SA) is an optimisation technique that balances efficiency and flexibility for finding acceptable solutions (Rao & Iyengar, 1994). Additionally, while genetic algorithms excel at exploring large search spaces, SA can achieve comparable results faster and with simpler implementations (Botsali, 2016), making SA better suited for the scope of this project.

SA explores a variety of solutions with a temperature variable that controls the acceptance of higher cost states, which is slowly reduced to find a global minimum. The following chapter elaborates on this concept. Moreover, the SA implementation will require detailed evaluation to ensure the implementation provides suitable output quality and acceptable compute times.

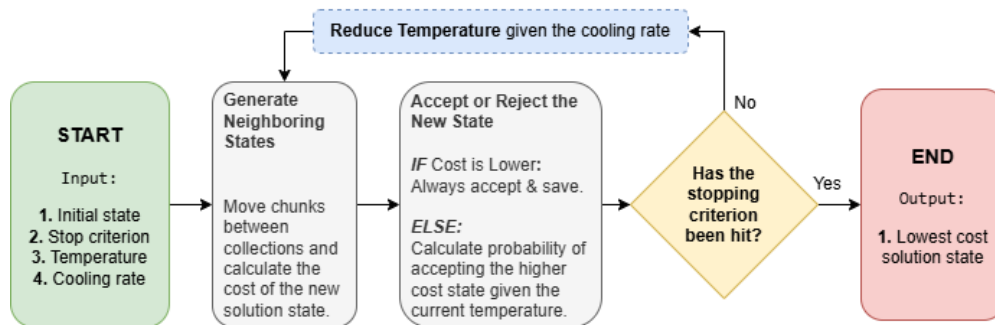


Fig. 2.3 – Simulated Annealing Flow Chart (Created with Draw.io)

## 2.2.4 Text Generation

Building upon the outlined set of collections defined by the previous step, this final process bridges the gap between structured specifications and generating natural text using LLMs.

### 2.2.4.1 Prompt Architecture

Users will be expected to customise their own prompts to generate training data aligned with their specific requirements as textual prompts directly impact the output quality. Hence, this needs to be tailored to individual needs (e.g. stylistic specifications or including examples).

Most notably, each template must incorporate metadata from each chunk within a collection through specialised placeholder variables that outline the subject matter (e.g. "Discuss {topic}"), state a clear sentiment orientation (e.g. "Express a {sentiment} opinion") and define strict word count limits (e.g. "Use exactly {word\_count} words"). This approach allows for the definition of highly specific instructions suited to the needs of different organisations, while still ensuring the constraints governed by the rulebook are still adhered to.

### 2.2.4.2 LLM Integration

The generation architecture will make use of externally hosted LLMs accessed through standardised APIs, ensuring that the framework remains adaptable to evolving technology.

The organisation or user will configure the connection to the LLM, either by hosting their own model or by relying on other providers. This modular API-centric design provides flexibility, enables seamless model substitution and also provides straightforward accessibility to third-party LLM infrastructure for users with limited computational resources.

Moreover, synthetic text generation at scale necessitates efficient request management to optimise throughput while maintaining quality control. Thus, the application must consider implementing these features to ensure the generation pipeline is robust and efficient while under varying workloads. This will help to ensure a stable user experience.

### 2.2.5 User Interface

To facilitate efficient interaction with the framework's capabilities, a graphical user interface (GUI) is essential, particularly for users without technical expertise. However, as it is not the primary focus of the project, it should leverage existing libraries to avoid unnecessary friction.

The interface must allow users to easily upload, validate and manage different rulebooks. Given a selected rulebook, users can generate a dataset structure and view its attributes before generating text for each collection. The GUI should also enable export of the final text.

## 2.3 Project Management

### 2.3.1 Requirements List

Building upon the discussion presented in this chapter and the project's aims discussed in the background research, a set of objectives and measurable criteria can be defined.

<i>Criteria</i>	<i>Functionality (key assessable requirements are <u>underlined</u>)</i>
<i>1. Rulebook Interface</i>	<ul style="list-style-type: none"><li>• Develop a data structure to store all aforementioned user-defined requirements in an organised manner.</li><li>• House each constraint set in a <u>single file</u> with a <u>non-technical means of easily adding or altering specific values</u>.</li></ul>
<i>2. Chunk Division</i>	<ul style="list-style-type: none"><li>• Implement an algorithm that uses a relevant subset of the rulebook parameters to outline a series of valid chunks.</li><li>• Employ <u>validation to ensure a logically possible and legal partition exists</u> which satisfies the provided criteria.</li></ul>
<i>3. Collection Forming</i>	<ul style="list-style-type: none"><li>• Design a sophisticated simulated annealing search algorithm that <u>groups chunks based on user-defined size ranges</u>.</li><li>• Employ relevant stopping criteria and optimise <u>processing to below a minute</u> for acceptable wait times.</li></ul>

4. Text Generation	<ul style="list-style-type: none"> <li>• Create a template engine for rendering placeholder values.</li> <li>• Apply <u>validation to reject the usage of invalid templates</u>.</li> <li>• Manage <u>concurrent LLM generation</u> using asynchronous requests implementing <u>error handling with state saving</u>.</li> </ul>
5. User Interface	<ul style="list-style-type: none"> <li>• Implement a <u>simple-to-use local web interface</u> with two pages:</li> <li>• <u>Rulebook Page</u>: Upload, delete and view rulebook data.</li> <li>• <u>Dataset Page</u>: Create and delete dataset structures. Visually analyse dataset profiles. Generate and export the final text.</li> </ul>

A diagram defining the complete pipeline with the flow of data is available in Appendix C: 1.1.

### 2.3.2 Agile Workflow

Given the scale of the project, an iterative approach to development will ensure the delivery of a solution if the project's timeline is not met. Thus, an agile sprint workflow will be applied.

Phase	Action Items
1. Sprint	<p><b>1a)</b> Create the rulebook only for topic-sentiment distributions and “word” mode.</p> <p><b>1b)</b> Validate for legal chunk partitions and randomly apply word counts.</p> <p><b>1c)</b> Implement a temporary POC backtracking algorithm for collection forming.</p>
2. Sprint	<p><b>2a)</b> Add “chunk” mode, chunk preferences and collection ranges to rulebook.</p> <p><b>2b)</b> Improve chunk partitioning to account for the new rulebook preferences.</p> <p><b>2c)</b> Design a simplified simulated annealing algorithm for grouping chunks.</p> <p><b>2d)</b> Implement the prompt rendering and API framework for LLM generation.</p>
3. Sprint	<p><b>3a)</b> Prepare statistics and visualisation functions for dataset analysis.</p> <p><b>3b)</b> Set up the GUI and develop both the rulebook and dataset pages.</p> <p><b>3c)</b> Create a method for exporting finalised datasets generated with the tool.</p>
4. Sprint	<p><b>4a)</b> Iterate upon the simulated annealing algorithm to improve efficiency.</p> <p><b>4b)</b> Clean up redundant files and testing code within the project's repository.</p> <p><b>4c)</b> Add code comments where necessary and write up formal documentation.</p>

### 2.3.3 Version Control

Adhering to standard software development methodologies, all modifications to the codebase will be tracked using Git. GitHub will host the remote repository together with basic issue tracking and documentation. After each sprint, a branch will be created to archive the repository state in order to track the codebase after each phase of development.

## **Chapter 3 – Implementation and Validation**

### **3.1 Technical Configuration**

#### **3.1.1 Programming Languages**

Python is used for the majority of the project's source code thanks to its flexibility and system compatibility. It is a language commonly used for the development of AI focused applications, stemming from its robust community support for external packages, including Pandas and NumPy for highly optimised mathematical operations (Sllaparasetty, 2024). Not only is this important considering algorithmic complexity of the pipeline and for futureproofing, but it also allows for easy integration with other Python codebases that focus on training NLP models.

#### **3.1.2 Imported Libraries**

This project required significant tailor-made functionality that could not be adequately met by a framework like LangChain. Therefore, the majority of the algorithmic design of the pipeline is custom built to allow for flexibility and transparency within the generation process. As such, the project does not involve a massive dependency on external libraries to handle core logic.

The main external library relied upon to expedite development was Streamlit (Streamlit, n.d.), which is an open-source lean GUI library built for creating simple and practical web apps that mainly focus on AI and data analytics. Furthermore, managing the behaviour and design of each page is handled entirely through Python, which helps to maintain a minimal tech-stack and limits points of failure compared to alternatives usually built with HTML and JavaScript.

### **3.2 Algorithmic Implementation**

#### **3.2.1 Rulebook Interface**

To allow for the simple manipulation and alteration of rulebooks the framework makes use of spreadsheet files that can be modified using Microsoft Excel (Excel, n.d.) or the open-source alternative LibreOffice Calc (Calc, n.d.). Additionally, providing users with an interface many are familiar with helps to keep the solution accessible while also accelerating development. Moreover, it allows for straightforward modifications to the rulebook object if required.

All the user requirements can be contained within a single ".xlsx" file, where the content and collection constraints each have a discrete sheet. Furthermore, these files contain extensive data validation functionality that is used to ensure appropriate user input for each table field.

A rulebook template is made available as a part of the codebase. Users are expected to edit values within each table while leaving all structural aspects unchanged. Following this, the OpenPyXL toolkit is used to extract values from the file. A final check is also performed to validate the integrity before the data is further processed (see Appendix C: 2.1.1 for testing).



	A	B	C	D	E	F	G
1	Personal Laptop		Collection Mode		word	Total	10000
2	TOPICS		SENTIMENT				
3	Topics Proportion Validity → SUM(Proportion) == 1		Sentiment Ratios Validity → UNIQUE(Validity) == {Valid}				VALID
4	Topic name	Proportion	Positive	Neutral	Negative	Total	Validity
5	Performance	0.20	0.50	0.30	0.20	1.00	VALID
6	Battery Life	0.15	0.40	0.40	0.20	1.00	VALID
7	Display Quality	0.12	0.45	0.35	0.20	1.00	VALID
8	Design & Build	0.10	0.55	0.30	0.15	1.00	VALID
9	Portability	0.10	0.50	0.30	0.20	1.00	VALID
10	Keyboard & Touchpad	0.08	0.40	0.40	0.20	1.00	VALID
11	Connectivity & Ports	0.07	0.35	0.40	0.25	1.00	VALID
12	Storage & Memory	0.08	0.45	0.35	0.20	1.00	VALID
13	Price & Value	0.07	0.30	0.40	0.30	1.00	VALID
14	Customer Support & Warranty	0.03	0.20	0.30	0.50	1.00	VALID

Fig 3.1 – Screenshot of the rulebook spreadsheet interface (full rulebook at Appendix C: 2.2.1)

### 3.2.2 Chunk Partitioning

This process creates chunks for each topic-sentiment pair from its allocated word/chunk budget. After checking for invalid rulebook parameters (e.g. allocated budget  $N$  cannot be partitioned into set  $S$ , where  $\min_i \leq c \leq \max_i$  for every  $c \in S$  and  $i$ -th topic), a concrete number is selected from the feasible range using a chunk count preference parameter.

Each chunk is first allocated the minimum word count using the word count size boundaries for its given topic. Given this set of chunks, the remaining word count ( $N - |S| * \min_i$ ) and the maximum amount a chunk can receive to stay within the limits ( $\max_i - \min_i$ ), the remaining words are then allocated across all the chunks using a Dirichlet distribution. A concentration parameter  $\alpha$ , obtained from the chunk variance preference in the rulebook, controls the size diversity of the resulting chunks, where higher  $\alpha$  values yield more uniform results.

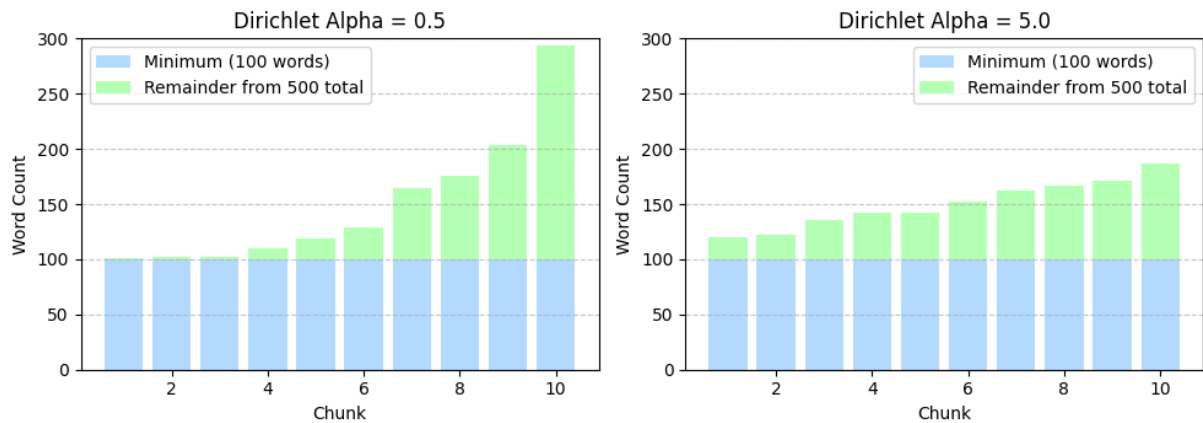


Fig. 3.2 – Remainder distribution affected by Dirichlet alpha parameter (Created with Matplotlib)

If the Dirichlet distribution leads to some chunks receiving a word count higher than their  $\max_i$ , the difference is redistributed to other chunks below the limit. This is always possible given the prior partitioning validation. The relevant tests can be viewed in Appendix C: 2.1.2.

### 3.2.3 Collection Forming

The employed simulated annealing algorithm represented the single most complex element in the generation pipeline. Ensuring the final implementation was both efficient and robust required several iterations and refinements to its architecture. The final design is covered below and the following chapter evaluates its performance, comparing all configurations.

#### 3.2.3.1 Overall Objective

The cost of a given state is defined by the divergence of the actual fractions from the target fractions across all size ranges in absolute terms. Additionally, to account for collections that are out-of-range (either below the minimum or above the maximum), two special ranges with target fractions of zero are used to categorise them during the search, ensuring that groups of chunks falling outside the specified size ranges are also factored into the cost.

Finding solutions that closely follow the constraints involves evaluating the cost of each state. While exploring the search space, SA will always accept moves resulting in a lower cost, but to avoid local minima (where better solutions are only accessible after making worse moves), SA can occasionally accept higher cost states. This decision is made using a temperature value, where higher temperatures have an increased likelihood of acceptance. The temperature is gradually decreased or “cooled” to converge towards the global minimum.

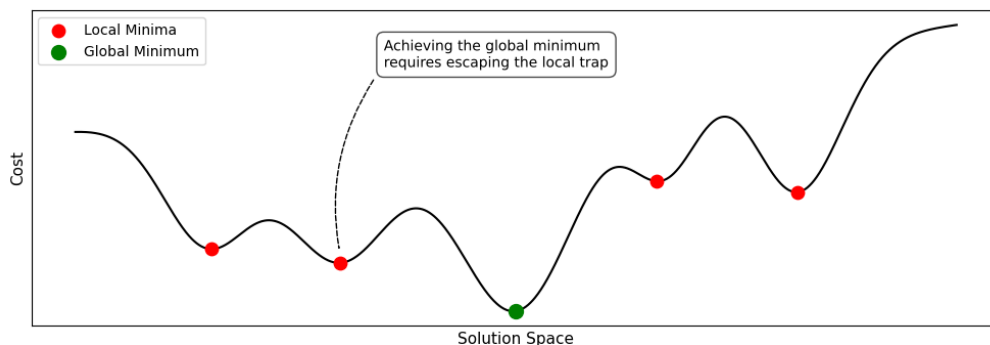


Fig 3.3 – Navigating local minima by considering higher cost states in SA (Created with Matplotlib)

#### 3.2.3.2 Code Implementation

Earlier attempts at implementing the SA algorithm manipulated with states as a simple list of collections, each containing its own set of chunks. Whenever a new neighbouring state was created, the search would perform a deep copy of the entire data structure, which proved to be a very time-consuming operation. Moreover, since most changes only impact a minority of the collections, continuously re-evaluating all size ranges resulted in redundant calculations, which would accumulate considering the need to obtain the cost after each iteration.

To improve the computational efficiency and to allow the search to consider a wider array of states in the same amount of time, the framework incorporates a specialised solution state object, which includes several functions to handle the efficient manipulation of collections.

In addition to a primary list containing all collections, the object also employs a series of lists with weak references for  $O(1)$  access to size range information. Only relevant parts of each list are updated when a move is performed, allowing for efficient chunk movements and cost calculations. Thanks to this, each change made by the search is now tracked and can easily be reverted to minimise the need for deep copies, as can be seen in the pseudocode below.

**Algorithm 1** Simulated Annealing for Collection Forming

---

**Input:** *InitialSolution, InitialTemperature, CoolingRate, MaxIterations*

**Output:** *BestSolution*

---

```
1:  CurrentSolution  $\leftarrow$  InitialSolution
2:  BestSolution  $\leftarrow$  DeepCopy(CurrentSolution)
3:  CurrentCost  $\leftarrow$  CalculateCost(CurrentSolution)
4:  BestCost  $\leftarrow$  CurrentCost
5:  Temperature  $\leftarrow$  InitialTemperature
6:  FOR Iteration  $\leftarrow$  1 to MaxIterations DO
7:      CurrentSolution, MoveInfo  $\leftarrow$  CreateNeighbouringState(CurrentSolution)
8:      NewCost  $\leftarrow$  CalculateCost(CurrentSolution)
9:       $\Delta$ Cost  $\leftarrow$  NewCost – CurrentCost
10:     IF ( $\Delta$ Cost < 0) or (Random(0,1) <  $\text{Exp}(-\Delta\text{Cost}/\text{Temperature})$ ) THEN
11:         CurrentCost  $\leftarrow$  NewCost
12:         IF NewCost < BestCost THEN
13:             BestSolution  $\leftarrow$  DeepCopy(CurrentSolution)
14:             BestCost  $\leftarrow$  NewCost
15:         END IF
16:     ELSE
17:         RevertMove(CurrentSolution, MoveInfo)
18:     END IF
19:     Temperature  $\leftarrow$  Temperature  $\times$  CoolingRate
20: END FOR
21: RETURN BestSolution
```

---

Additionally, to further refine the search, three heuristic functions are employed to help with selecting neighbouring states. During each iteration, one is chosen at random to propose a move that is likely to lead towards a reduction in overall cost. These include focusing efforts on overpopulated size ranges, swapping larger chunks with smaller ones and strategically splitting oversized collections. A more detailed explanation is available in Appendix C: 2.1.3.

Moreover, an initial approach of allocating each chunk to its own collection as a starting state commonly incurred a high cost. Consequently, this significantly burdened the SA algorithm.

Therefore, the final implementation employs a two-phase approach. Before optimising with SA, a greedy algorithm is used to deterministically generate a sufficiently low-cost starting solution. As is illustrated below, the algorithm first sorts the chunks by topic frequency and then assigns each chunk with the most frequent topic to its own collection, respecting the hard constraint. Following this, it estimates the number of collections per size range and allocates the remaining chunks to existing or new collections to best match this distribution.

Both the SA and greedy algorithms were tested appropriately (detailed in Appendix C: 2.1.4).

---

**Algorithm 2** Greedy Algorithm for Initial Solution

---

**Input:** *Chunks, SizeRanges*

**Output:** *Solution*

---

```
1:  Solution  $\leftarrow$  InitializeSolutionStructure(Chunks, SizeRanges)
2:  SortedChunks  $\leftarrow$  SortChunksByTopicFrequencyAndSize(Chunks)
3:  MostFrequentTopic  $\leftarrow$  SortedChunks[0].GetChunkTopic()
4:  FOR Chunk IN SortedChunks DO
5:      IF Chunk.GetChunkTopic() = MostFrequentTopic THEN
6:          Solution.CreateNewCollectionWith(Chunk)
7:          SortedChunks  $\leftarrow$  SortedChunks \ Chunk
8:      END IF
9:  END FOR
10: EstimatedCounts  $\leftarrow$  EstimateCollectionsPerSizeRange(Chunks, SizeRanges)
11: MaxRange  $\leftarrow$  GetMaxSizeRange(SizeRanges)
12: FOR Chunk IN SortedChunks DO
13:     EligibleCollections  $\leftarrow$  Solution.GetOptionsForChunkBelowRange(Chunk, MaxRange)
14:     IF Len(EligibleCollections) > 0 THEN
15:         SelectedCollection  $\leftarrow$  Solution.AddToBestCollection(Chunk, EligibleCollections)
16:         IF IsInOrAboveRange(SelectedCollection, MaxRange) THEN
17:             EstimatedCounts[MaxRange]  $\leftarrow$  EstimatedCounts[MaxRange] - 1
18:             IF EstimatedCounts[MaxRange] = 0 THEN
19:                 MaxRange  $\leftarrow$  GetSizeRangeBelow(MaxRange)
20:             END IF
21:         END IF
22:     ELSE
23:         Solution.CreateNewCollectionWith(Chunk)
24:     END IF
25: END FOR
26: RETURN Solution
```

---

### 3.2.4 Text Generation

The core generational ability of individual LLM models is directly impacted by the quality of the prompts used to guide their output. Accordingly, users will always be expected to include:

- Detailed instructional explanation that establishes context and output expectations.
- Structural guidelines for organising the generated output text in a standard format.
- Optional stylistic directives to ensure tonal consistency or variety within the dataset.

Prompt templates with these considerations in place are available as part of the solution, but users are still able to fully customise these prompts, as long as they include all placeholder variables (which are later replaced with values from the collection forming process).

Each template is its own HTML file with instructions defined in natural language. This is to allow Jinja2 (Jinja, n.d.), an efficient templating engine, to accept a set of variables together with an HTML template to reliably render the final prompt. Prompt templates that are missing the required placeholder values cause rendering errors, which are caught by the system and handled gracefully by informing the user to restructure the file according to specification.

Moreover, it is also crucial to construct the generation pipeline in a way that best empowers LLMs to adhere to the dataset structure. For this reason, two separate prompting techniques are made available to suit the needs of various dataset requirements and model capabilities:

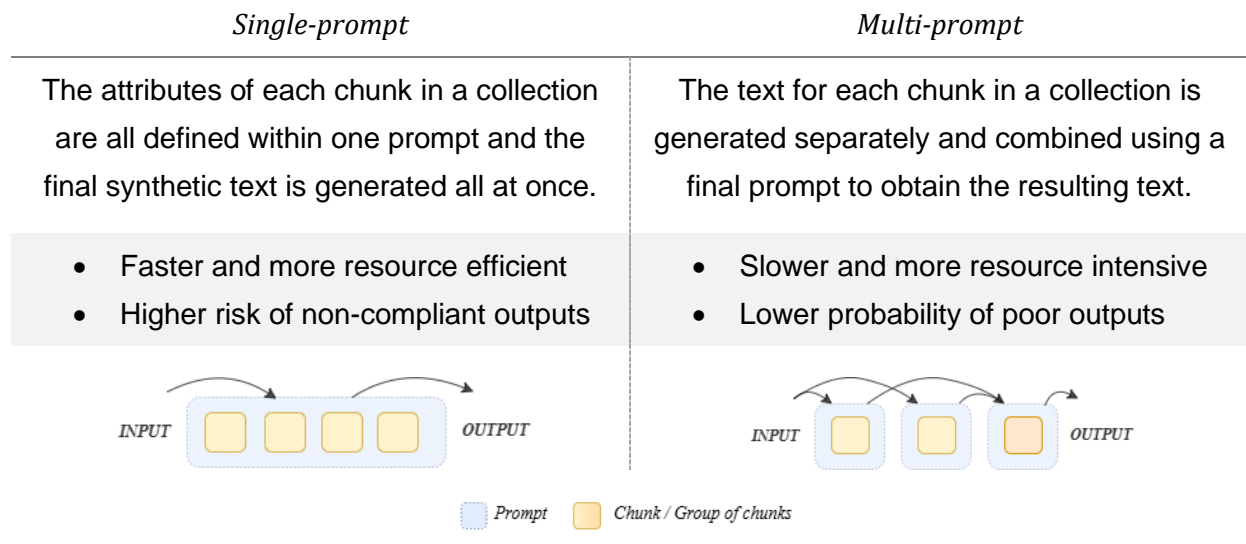


Fig 3.4 – Comparison of prompting approaches with multiple chunks (Created with Draw.io)

All requests are sent to the LLM via a user-defined host and are processed in parallel. For the multi-prompt approach, each discrete chunk is generated first followed by a connecting prompt to finalise the full collection. Every response is checked and unsuccessful attempts are sent again after a short pause, with a user setting stating the maximum number of retries. Furthermore, any unsuccessful requests do not affect the others. All successfully generated text is saved and the user is simply presented with the relevant errors.

### 3.2.5 User Interface

The GUI is handled by Streamlit and can be interacted with locally through a web browser or hosted as a web application. However, providers would need to configure it for deployment.

The application stores and manipulates data using JSON files due to their user-friendliness, portability and ease of reading/writing with Python. Legal rulebooks provided by the user are automatically converted and any generated datasets are always stored as a JSON. The files are also always validated before any data is accessed to prevent unexpected runtime errors.

The interface is built solely using the toolkit's core components, which helps to ensure a level of stability and integrity. Consequently, thorough testing with third-party tools was considered unnecessary. However, user testing was conducted to evaluate the usability and navigation of the GUI. The results of this testing will be discussed in detail within the next chapter.

#### 3.2.5.1 Rulebook Page

A simple form is provided through which users can upload rulebooks, either in a spreadsheet format or directly as a JSON file. If structured correctly, it will be stored in a relevant directory and selected for preview. Otherwise, the user will be notified by an informative banner. Each uploaded rulebook can also be viewed in detail, renamed, or deleted within the same page.

Screenshots of the full functionality available on this page are available in Appendix C: 2.2.2.

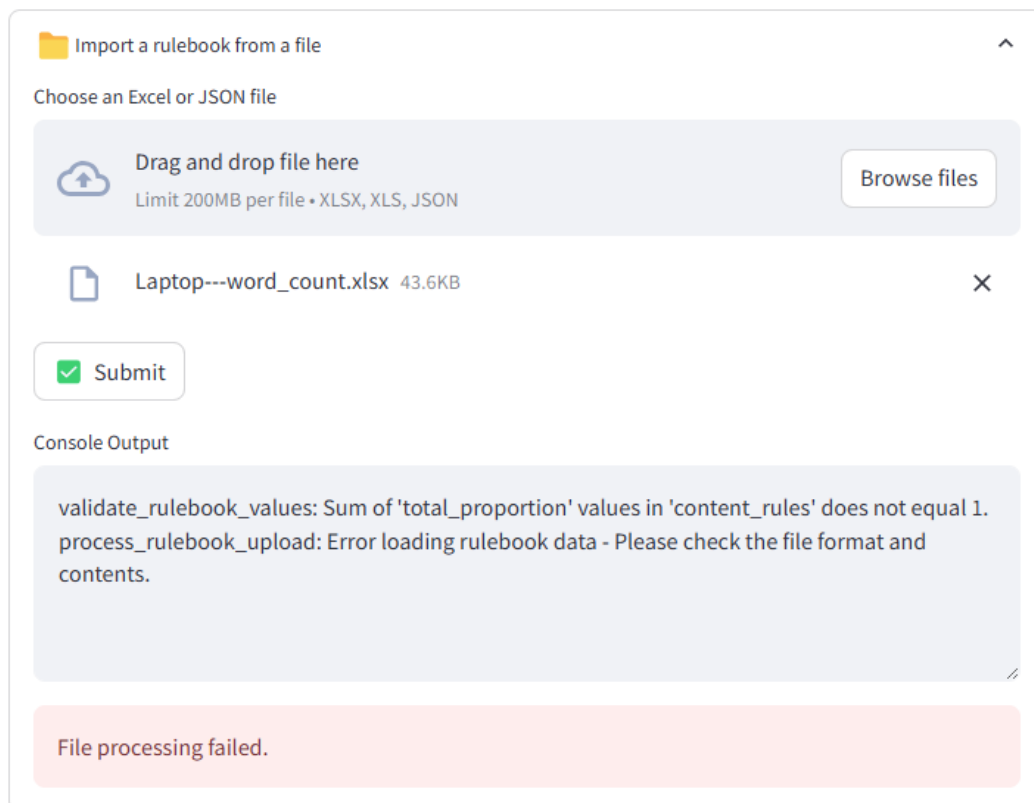


Fig 3.5 – Rulebook upload component after invalid submission (Screenshot of Streamlit)

### 3.2.5.2 Dataset page

Any valid rulebook stored within the application can be used to generate dataset structures. If any issues arise during this process, such as partitioning problems, users are informed with a message similar to the example above. Navigating between each generated dataset is done using a selector component. In addition to renaming and deletion functionality, there are five tabs the user can switch between to further interact with the chosen dataset. These are:

<i>View</i>	<i>Description</i>
<i>1. Dataset Metrics</i>	Users can view a full analysis of the generated dataset including: <ul style="list-style-type: none"><li>• <b>Basic metrics</b> on dataset size and amount of generated text.</li><li>• <b>Visualisations</b> illustrating different aspects of the dataset:<ul style="list-style-type: none"><li>○ Collection size distribution by chunk or word count.</li><li>○ Topic coverage distribution by chunk or word count.</li><li>○ Overall sentiment distribution by chunk or word count.</li><li>○ Word count distribution by chunk with outliers.</li></ul></li><li>• <b>Conformity metrics</b> showing % match with initial requirements.</li></ul>
<i>2. Collections Table</i>	All collections can be viewed in a tabular format with filtering options: <ul style="list-style-type: none"><li>• <b>Table columns:</b> ID, Number of chunks, Total word count, List of chunks, Word count distribution across all chunks, Has-text flag.</li><li>• <b>Filter options:</b> Word count range, Chunk count range, Filter by topic, Filter by sentiment, Has-text flag.</li></ul>
<i>3. Collection Viewer</i>	Exhibits details of any collection and enables individual text generation: <ul style="list-style-type: none"><li>• <b>Collection selector</b> using IDs found in the Collections table.</li><li>• <b>Text generation</b> capabilities including fields for selecting LLM models, generation strategy and user-defined prompt templates.</li><li>• <b>Chunk details</b> displaying its profile and generated text, if any.</li></ul>
<i>4. Text Generator</i>	Provides functionality for bulk generation using the dataset structure: <ul style="list-style-type: none"><li>• <b>Completeness metrics</b> measuring text available for generation.</li><li>• <b>Text generation</b> functionality, similar to the component above.</li><li>• <b>Status report</b> after bulk generation stating any response errors.</li></ul>
<i>5. Export Data</i>	Enables the export of data stored and generated with the program: <ul style="list-style-type: none"><li>• <b>JSON download</b> containing the dataset metadata structure.</li><li>• <b>Text file download</b> of all the generated collection text units.</li></ul>

Validation and styling are consistent with all the components for a smooth user experience. See Appendix C: 2.2.3 for images detailing all the dataset page components listed above.

## **Chapter 4 – Results, Evaluation and Discussion**

### **4.1 Evaluation**

#### **4.1.1 Output Text Quality**

To assess the unique functionality outlined in the methodology chapter, it's important to first consider the project's overall context. Notably, the quality of the text output and adherence to the constraints are significantly dependent upon prompt structure and the capabilities of the LLM. Given that the user manages these factors to allow for contextual flexibility, the impact of these variables will not be evaluated. However, development and user testing showed that modern LLMs can effectively generate realistic dataset elements that comply with a list of chunks provided in their prompt. An example of a successful output can be seen in Appendix C: 3.1.1, demonstrating the product's ability to fully serve its intended purpose. This shows that the fundamental goal of the project has been achieved and that the solution works.

Nevertheless, there are scenarios where the product may not function as intended. Several mainstream LLM models have filters or have been trained to detect inappropriate requests and can decline to respond as instructed. Appendix C: 3.1.2 shows an example of OpenAI's GPT-4o refusing to generate a realistic product review. However, these issues can easily be avoided by fine-tuning an LLM or by using explicit prompting (e.g. mentioning that the output is used to help train language models – as was the case with the successful example above).

#### **4.1.2 Performance Metrics**

Two factors significantly impact the overall speed of the pipeline. The first is the collection forming process, which is done locally and is fundamentally an optimisation problem. Its performance can be measured by execution time and final solution cost, allowing for an objective evaluation. The second aspect, LLM request handling, is reliant on external factors like the infrastructure used, network bandwidth and rate limits. Therefore, this section of the evaluation will omit this functionality and focus on assessing the collection forming process.

A comprehensive evaluation script was developed to measure the performance of different configurations for the SA algorithm. Although the final SA implementation accepts additional tuning parameters, this evaluation only compares different iteration limits, cooling rates and starting solutions. Also, all tests used the same rulebook with 30000 words, which was divided into 792 chunks, given the remaining constraints outlined in the rulebook.

The scatter plot below compares these configurations, where lower deviation scores and execution times are better. Green points show runs with each chunk starting in its own collection while blue points use the greedy solution. All configurations were run 5 times, where the resulting data was averaged to help mitigate any external influences. Additionally, each run would automatically stop after 1000 iterations without improvement to save time.



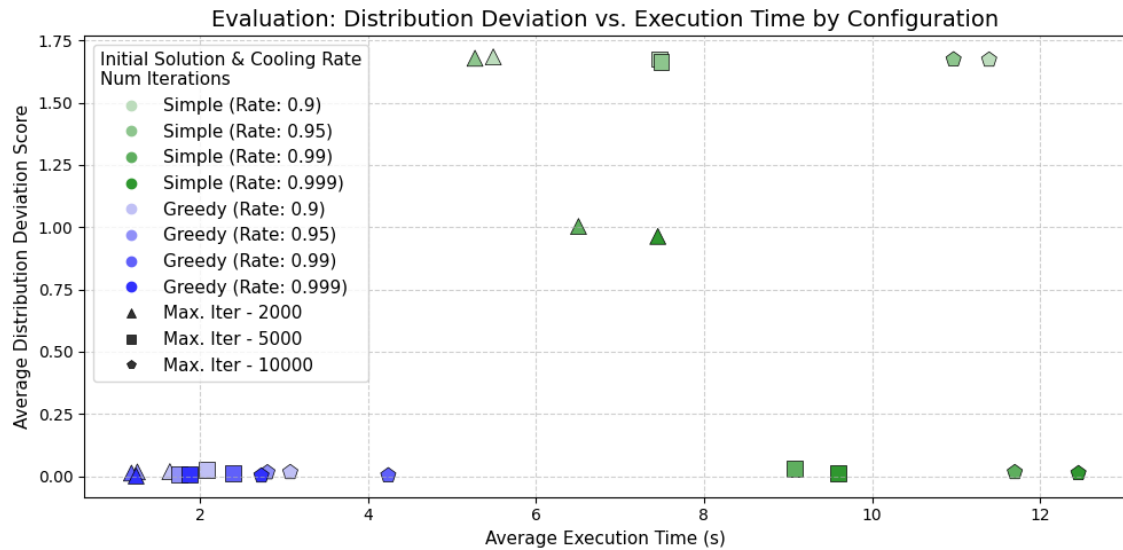


Fig 4.1 – Different SA parameters and their effect on performance (created with Matplotlib)

The scatter plot clearly shows that the most significant factor impacting performance is the initial solution used. All greedy configurations obtained a near-zero final cost within a few seconds, where lower iteration limits paired with higher cooling rates offered slightly more favourable results. In contrast, many configurations using the simplified solution struggled to reach an optimal state. Runs with lower cooling rates and iteration limits could not explore the search space sufficiently and escape local minima. However, given higher cooling rates and more time to evaluate different states, some configurations starting with the simple solution were able to obtain near-optimum states at the expense of longer compute times. All of the tests were conducted using the same hardware. Hence, the results may differ on other systems. The table of results containing all configurations can be viewed in Appendix C: 3.2.

It is also worth noting the difference in performance between the SA solution implemented during the second sprint of this project and the final optimised version designed in the fourth sprint, which was used in the testing above. The two figures below compare both versions using the greedy solution, the same rulebook and an iteration limit of 10000. A cooling rate of 0.99 was selected. This was the value under which the older version performed best.

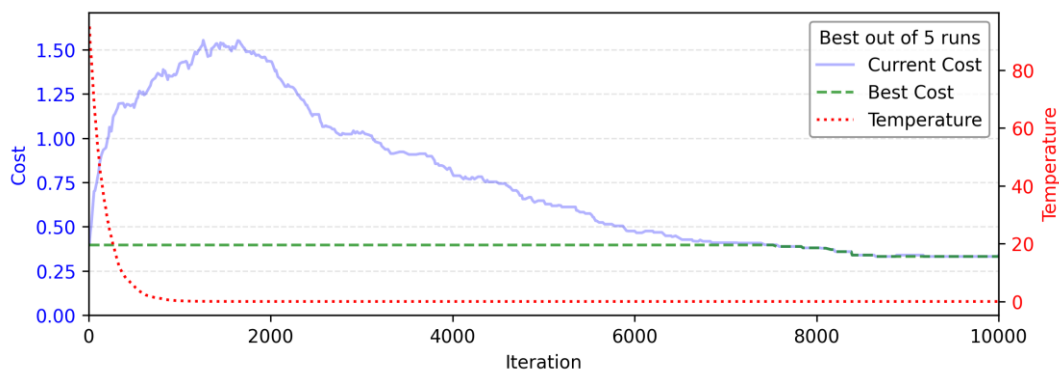


Fig. 4.2 – Cost evolution over time for the older SA implementation (created with Matplotlib)

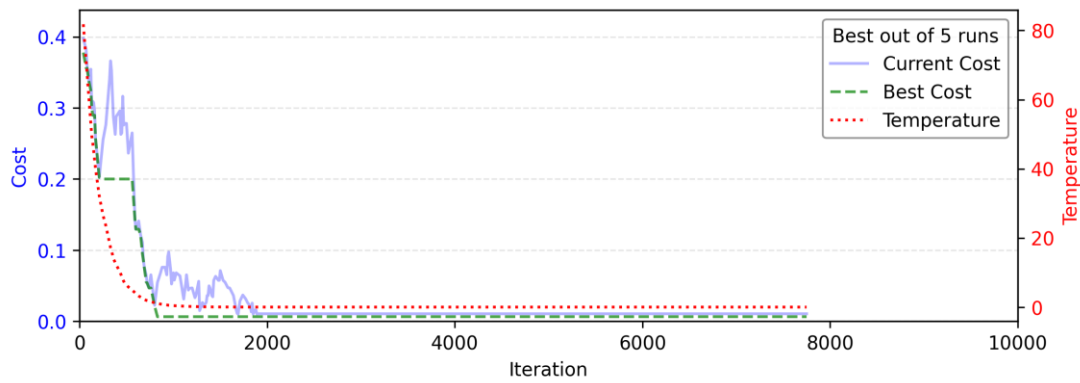


Fig. 4.3 – Cost evolution over time for the optimised SA implementation (created with Matplotlib)

The design of the SA implementation followed the agile development approach, whereby a minimum viable product was first created and then iterated upon. The older version (fig. 4.2) does not employ any heuristics and makes moves at random. Hence, it cannot explore any strategic moves to build upon the greedy solution, which explains the cost increase at the start (when more expensive and experimental moves are likely to be accepted). Conversely, the optimised version employs three heuristics outlined in Appendix C: 2.1.3, which help to reduce cost even in the early stages of the search and move towards a near-optimal solution.

Moreover, not only does the newer implementation succeed in discovering a near-zero cost solution but it is also significantly faster (2.07 vs. 39.86 seconds). Moreover, it halted due to a lack of improvement (a mechanism explained previously), implying that comparable results were feasible in an even shorter period of time. These time savings can be attributed to the solution structure class used by the newer SA implementation, subsequently improving the user experience. The full table comparing the full results can be found in Appendix C: 3.3.

These findings show that the method of implementing the SA algorithm drastically impacts its performance. Utilising a two-phase approach with the initial greedy solution yields more accurate results. Furthermore, due to SA's iterative nature, adding heuristics to guide move generation as well as ensuring efficient data handling contributes to faster overall processing.

#### 4.1.3 Usability and Integrity

A sample of 5 students familiar with AI and its data dependence was invited to take part in a study to evaluate the usability of the final product. The small sample size reflects the limited capacity for detailed GUI optimisation within the project's timeline. The primary objective was to validate and test the essential utilities of the GUI, whilst identifying areas for improvement.

Each candidate was first provided with an information sheet to help ascertain the context of the study and how their responses would be handled. All participants signed a consent form before taking part, ensuring the right to withdrawal and anonymity. To guarantee a consistent environment, all participants used the same computer with the application already running.

Each participant was provided with a task list which outlined the set of actions they needed to complete before providing feedback. The order of tasks was designed to lead the participants through the complete pipeline, starting with slight rulebook modifications, followed by the use of the Streamlit app and concluding with the export of the generated text. Once finished, each participant answered 8 questions evaluating each step of the procedure. All responses were provided on a 1-5 scale, allowing for a simple comparison between all of the opinions. All the materials prepared and used to conduct this study can be found in Appendix C: 3.4-7.

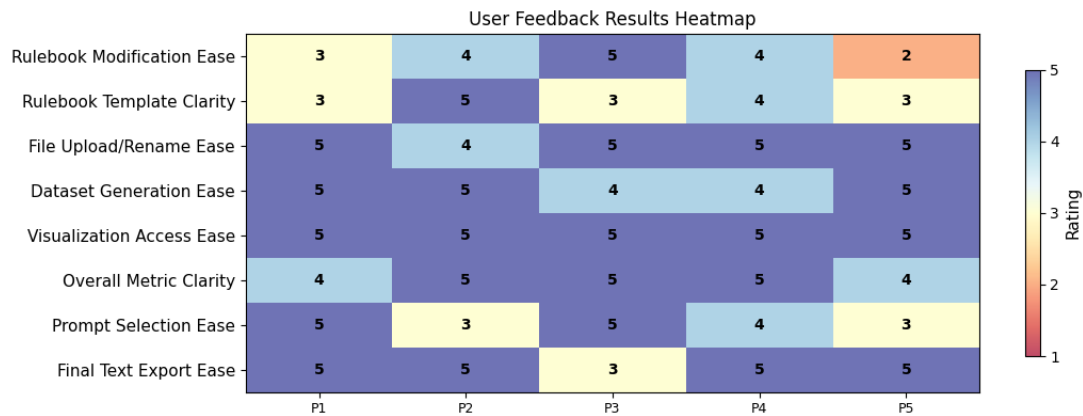


Fig 4.4 – Participant ratings on individual steps within the pipeline (created with Matplotlib)

The results of the study are visualised on the heatmap above, where higher scores indicate greater ease or clarity for the given task and each column corresponds to a set of responses from each participant. Overall, the framework received positive feedback, which was mainly attributed to file handling, dataset generation, visualisation metrics and the final text export functionality. Importantly, all attempts were successful without any runtime errors.

However, the heatmap also reveals areas for improvement. Modifying the rulebook template proved to be challenging for some participants and the Excel interface was also perceived as underdeveloped. This could be iterated upon and included directly within the Streamlit app together with better parameter explanations for a more streamlined experience. Moreover, the prompt selection module also led to confusion. Similarly, allowing users to view and edit prompt templates instead of only selecting from a list of HTML files could improve the core usability and integrate more functionality within the web app, further reducing fragmentation.

## 4.2 Conclusion

After critically evaluating the performance of the final SA implementation and analysing user feedback concerning the interface for the generation pipeline, it is apparent that the solution works and the most challenging aspects of the project are developed effectively. A thorough analysis was essential for these components, as their implementation approach considerably affects the usability of the final solution. Conversely, the remaining functionality only required standard testing, referenced throughout the previous chapter, due to their lesser impact.

Furthermore, by reviewing the requirements list outlined in the methodology chapter, which directly builds upon the set of foundational aims this project established in the background research, it is clear that these goals have been met by the final product. Specifically:

1. **The Rulebook Interface** is contained within a single Excel workbook file for easy manipulation and cross-platform compatibility. Also, by design, it is easily modifiable.
2. **The Chunk Division** process accepts a set of preference parameters and correctly allocates word counts from a budget, while rejecting requests that cannot be satisfied.
3. **The Collection Forming** process effectively groups all chunks within desired size ranges. The tests above involving 30000 words found satisfactory solutions in roughly 2 seconds, suggesting that greater demands could be met within a minute.
4. **The Text Generation** system is capable of handling hundreds of requests in parallel, as was proven during the user testing. There is robust error checking in place and the employed templating engine validates all custom prompts before rendering them.
5. **The User Interface** is functional and easy to navigate for most users, as was shown during the participant study. The graphical web interface can be operated locally or hosted separately and the application features both pages, which are complete.

Together these provide a robust framework that not only fulfils the original objectives but also empowers users with a streamlined solution for generating structured synthetic text datasets.

### 4.3 Future Work

There exist several opportunities to further refine the original concept. The most rudimentary enhancements could involve integrating all rulebook and prompt template manipulation within the Streamlit GUI, as was addressed in the user feedback. Moreover, the constraints defined within the rulebook could be expanded upon by including example formatting, stylistic preferences or additional sentiments. Furthermore, allowing users to amend individual chunks or collections after the dataset structure has been created by the pipeline would provide better content control and reduce the need for chunk/collection re-creation.

Furthermore, considering the expensive nature of utilising LLMs to generate text at scale, supplementary metrics could be incorporated to measure the number of tokens processed by each request. This data could be analysed to help users make predictions regarding the cost of generating text with a given dataset structure, helping to reduce training expenditure.

Nevertheless, the capabilities of the framework can be further extended beyond the original aims of the project, either by adding new functionality or by applying it to different contexts.

### **4.3.1 Output Validation**

A key feature users could greatly benefit from would be the inclusion of automated validation mechanisms to analyse the generated text, as there is currently no systematic method in place to ensure that the responses from the LLMs are valid and adhere to the requirements.

As was discussed in the initial chapter of this report, the quality of the training data directly impacts the performance of the model trained on it. While effective training data is attainable using an appropriate LLM with explicit prompting, this new technology is still inconsistent and prone to generating flawed responses. Therefore, employing a separate NLP model to verify any generated text could help users strengthen their datasets by flagging inaccurate results.

There are various approaches that could be applied depending on the degree of automation and considerations required. A straightforward method of validation could employ a separate LLM to compare the synthetic output with the specifications stated in the prompt. Alternate approaches may rely upon specialised NLP tools to assess the sentiment orientation or to determine whether certain stylistic features have been included in the final text. Moreover, since it is likely that some collections will have similar characteristics, additional refinement by regulating repeated ideas, phrases or words could further enhance the data's authenticity.

### **4.3.2 Alternative Applications**

The framework can be adapted to generate highly structured text corpora outside the scope of customer feedback. A noteworthy application would be to create synthetic datasets to help train text summariser models. This field of NLP has been studied extensively due to its ability to automate a vast array of business practices. Nonetheless, cutting-edge models still remain subject to errors and hallucinations, limiting their benefits in many cases. (Tang et al., 2023). However, by updating the project's pipeline to focus on factually correct datasets, where the rulebook serves as the ground truth, developers could build specialised summariser models that perform better in context-specific tasks compared to general-purpose alternatives.

On a similar note, there is an increasing need to develop models capable of determining if a passage of text is AI-generated or human-written. As mentioned in the background research, the demand for such models in the e-commerce sector is growing, as retailers aim to crack down on the influx of fake reviews (Poojary, 2024). However, other platforms can also seek to benefit from such tools. These include various communication services, like social media or messaging clients, as they are responsible for protecting their users. Consequently, such platforms will soon require more comprehensive tools to identify and filter out synthetically generated disinformation, spam or malicious content. To help fight these issues, the pipeline can be adapted to effectively generate large quantities of diverse AI-generated examples to aid in training models capable of detecting fake comments, phishing scams or other threats.

Ultimately, this framework positions itself as an adaptable baseline for various NLP projects.

## List of References

- Aggarwal, K., Jin, H., & Ahmad, A. (2022). *Entity-controlled synthetic text generation using contextual question and answering with pre-trained language models*. <https://www.amazon.science/publications/Entity-Controlled-Synthetic-Text-Generation-Using-Contextual-Question-and-Answering-with-Pre-Trained-Language-Models>.
- Ahn, J., & Khosmood, F. (2022). *Evaluation of Automatic Text Summarisation using Synthetic Facts*.
- Amrish, T., & Shwetank, A. (2024). Comparative analysis of manual and annotations for crowd assessment and classification using artificial intelligence. *Data Science and Management*. <https://doi.org/10.1016/j.dsm.2024.04.001>
- Avery, J. (2025, January 30). The Impact of LLMs on Search and Your Brand | Further. <https://www.gofurther.com/blog/the-impact-of-llms-on-search-and-your-brand>.
- Aysha, A. (2023, August 31). Generate synthetic text data - MOSTLY AI. <https://mostly.ai/blog/synthetic-text-data>.
- Benram, G. (2024, February 10). Understanding the cost of Large Language Models (LLMs). *Tensorops.ai*. <https://www.tensorops.ai/post/understanding-the-cost-of-large-language-models-llms>
- Berenstein, D., Diaz, S. H., Aguirre, L., Vila, D., Vi, A., & Burtenshaw, B. (2024). Introducing the Synthetic Data Generator - Build Datasets with Natural Language. In <https://huggingface.co/blog/synthetic-data-generator>.
- Blum, C., Hemmelmayr, V., Hernández, H., & Schmid, V. (2010). *Hybrid Algorithms for the Variable Sized Bin Packing Problem*.
- Botsali, A. R. (2016). COMPARISON OF SIMULATED ANNEALING AND GENETIC ALGORITHM APPROACHES ON INTEGRATED PROCESS ROUTING AND SCHEDULING PROBLEM. *International Journal of Intelligent Systems and Applications in Engineering*, 4, 101–104. <https://doi.org/10.18201/ijisae.267358>
- Bronsdon, C. (2024, November 18). Comparing LLMs and NLP Models: What You Need to Know - Galileo AI. <https://www.galileo.ai/blog/comparing-llms-and-nlp-models-what-you-need-to-know>.
- Calc, L. (n.d.). *LibreOffice Calc - Documentation*. <https://www.libreoffice.org/discover/calc/>.
- Duarte, A. V., Zhao, X., Oliveira, A. L., & Li, L. (2024). DE-COP: Detecting Copyrighted Content in Language Models Training Data. *Proceedings of Machine Learning Research*, 235, 11940–11956.

- Excel, M. (n.d.). *Excel - help & learning*. <https://support.microsoft.com/en-us/excel>.
- Ghaisas, S., & Singhal, A. (2024). *Dealing with Data for RE: Mitigating Challenges while using NLP and Generative AI*.
- Graziani, M., Foncubierta, A., Christofidellis, D., Espejo-Morales, I., Molnar, M., Alberts, M., Manica, M., & Born, J. (2025). *We Need Improved Data Curation and Attribution in AI for Scientific Discovery*.
- Guo, Y., Guo, M., Su, J., Yang, Z., Zhu, M., Li, H., Qiu, M., & Liu, S. S. (2024). *Bias in Large Language Models: Origin, Evaluation, and Mitigation*.
- Hao, S., Han, W., Jiang, T., Li, Y., Wu, H., Zhong, C., Zhou, Z., & Tang, H. (2024). *Synthetic Data in AI: Challenges, Applications, and Ethical Implications*.
- He, X., Nassar, I., Kiros, J., Haffari, G., & Norouzi, M. (2022). Generate, Annotate, and Learn: NLP with Synthetic Text. *Transactions of the Association for Computational Linguistics*, 10, 826–842. [https://doi.org/10.1162/tacl\\_a\\_00492](https://doi.org/10.1162/tacl_a_00492)
- Hu, M., & Liu, B. (2004). Mining and summarising customer reviews. *KDD-2004 - Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 168–177. <https://doi.org/10.1145/1014052.1014073>
- IBM, I. (2023, November 2). *What Are Large Language Models (LLMs)? | IBM*. <https://www.ibm.com/think/topics/large-language-models>.
- IMDB, I. (2024). *IMDb Data Files Download*. <https://datasets.imdbws.com/>.
- Jinja, J. (n.d.). Jinja. In <https://jinja.palletsprojects.com/en/stable/>.
- Johnson, D. S., Demers, T., A., Ullman, J. D., Gareyi, M. R., & Graham, R. L. (1974). *WORST-CASE PERFORMANCE BOUNDS FOR SIMPLE ONE-DIMENSIONAL PACKING ALGORITHMS\** (Vol. 3, Issue 4).
- Kasner, Z., & Dusek, O. (2024). Beyond Traditional Benchmarks: Analyzing Behaviors of Open LLMs on Data-to-Text Generation. *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 12045–12072. <https://doi.org/10.18653/v1/2024.acl-long.651>
- Keshavarzi, A., Qi, J., Hollink, L., Tjong Kim Sang, E., & Ceolin, D. (2024). Investigating the Usefulness of Product Reviews Through Bipolar Argumentation Frameworks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 14629 LNCS, 296–308. [https://doi.org/10.1007/978-3-031-62362-2\\_21](https://doi.org/10.1007/978-3-031-62362-2_21)
- Kim Amplayo, R., Brazinskas, A., Suhara, Y., Wang, X., & Liu, B. (2022). Beyond Opinion Mining: Summarising Opinions of Customer Reviews. *SIGIR 2022 - Proceedings of the*

- 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 3447–3450. <https://doi.org/10.1145/3477495.3532676>
- Koneva, N., Navarro, A. L. G., Sánchez-Macián, A., Hernández, J. A., Zukerman, M., & de Dios, Ó. G. (2025). *Introducing Large Language Models as the Next Challenging Internet Traffic Source*.
- Korkankar, P. D., Abranches, A., Bhagat, P., & Pawar, J. (2024, December). *Aspect-based Summaries from Online Product Reviews: A Comparative Study using various LLMs*. <https://aclanthology.org/2024.lcon-1.65/>.
- LangChain, L. (n.d.). LangChain. In <https://www.langchain.com/>.
- Lu, Y., Shen, M., Wang, H., Wang, X., van Rechem, C., Fu, T., & Wei, W. (2023). *Machine Learning for Synthetic Data Generation: A Review*.
- Malik, N., & Bilal, M. (2024). Natural language processing for analyzing online customer reviews: a survey, taxonomy, and open research challenges. *PeerJ Computer Science*, 10. <https://doi.org/10.7717/PEERJ-CS.2203>
- McAuley, J. (2023). Amazon Product Reviews. In [https://cseweb.ucsd.edu/~jmcauley/datasets.html#amazon\\_reviews](https://cseweb.ucsd.edu/~jmcauley/datasets.html#amazon_reviews). Association for Computing Machinery, Inc.
- Mehrafarin, H., Rajaei, S., & Pilehvar, M. T. (2022). On the Importance of Data Size in Probing Fine-tuned Models. *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 228–238. <https://doi.org/10.18653/v1/2022.findings-acl.20>
- Mehta, S. (2024, July 30). Enhancing Text Data Quality: A Guide to Detecting Issues with Cleanlab. <https://adasci.org/enhancing-text-data-quality-a-guide-to-detecting-issues-with-cleanlab/>.
- Mezzi, E., Mertzani, A., Manis, M. P., Lilova, S., Vadivoulis, N., Gatirdakis, S., Roussou, S., & Hmede, R. (2025). *Who Owns the Output? Bridging Law and Technology in LLMs Attribution*.
- Nadas, M., Diosan, L., & Tomescu, A. (2025). *Synthetic Data Generation Using Large Language Models: Advances in Text and Code*.
- Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N., & Mian, A. (2023). *A Comprehensive Overview of Large Language Models*.
- Otten, N. Van. (2023, September 19). Multilingual NLP Made Simple — Challenges, Solutions & The Future. <https://spotintelligence.com/2023/09/19/Multilingual-Nlp-Made-Simple-Challenges-Solutions-the-Future/>.



- Paget, S. (2025, February 27). Historical Trends in Consumer Review Behavior: Shifts in Social Awareness - BrightLocal. <https://www.brightlocal.com/research/consumer-reviews-historical-trends/>.
- Park, S., Chung, S., & Lee, S. (2019). The effects of online product reviews on sales performance: Focusing on number, extremity, and length. *Journal of Distribution Science*, 17, 85–94. <https://doi.org/10.15722/JDS.17.5.201905.85>
- Patibandla, B. T. (2024, September 14). *LLMs & Resource Management*. <https://medium.com/@patibandlagus15002/llms-resource-management-38e0e3150287>.
- Poojary, K. K. (2024, January). *Deciphering Deception - Detecting Fake Review using NLP by analysis of stylistic, sentiment-based, and semantic features*. <https://esource.dbs.ie/server/api/core/bitstreams/f5ede0ac-daec-41d1-8cb5-407961d740df/content>.
- Pope, A. (2024, April 10). NYT v. OpenAI: The Times's About-Face - Harvard Law Review. <https://harvardlawreview.org/blog/2024/04/nyt-v-openai-the-times-about-face/>.
- PyTest, P. (n.d.). *pytest: helps you write better programs*. <https://docs.pytest.org/en/stable/>.
- Rao, R. L., & Iyengar, S. S. (1994). Bin-packing by simulated annealing. *Computers and Mathematics with Applications*, 27, 71–82. [https://doi.org/10.1016/0898-1221\(94\)90077-9](https://doi.org/10.1016/0898-1221(94)90077-9)
- Rojo-Echeburúa, A. (2024, November 14). Small Language Models: A Guide With Examples. <https://www.datacamp.com/blog/small-language-models>.
- Segbroek, M. Van, Corneil, D., Nathawani, D., Meyer, Y., & Tramel, E. (2025, February 25). Building Synthetic Datasets with Reasoning Traces Using Gretel Navigator. <https://gretel.ai/blog/synthetic-datasets-with-reasoning-traces-using-gretel-navigator>.
- Siledar, T., Nath, S., Muddu, S., Rangaraju, R., Nath, S., Bhattacharyya, P., Banerjee, S., Patil, A., Singh, S., Chelliah, M., & Garera, N. (2024). One Prompt To Rule Them All: LLMs for Opinion Summary Evaluation. *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 12119–12134. <https://doi.org/10.18653/v1/2024.acl-long.655>
- Sllaparasetty, V. (2024, June 26). *Top 10 AI Programming Languages: A Beginner's Guide to Getting Started*. <https://www.datacamp.com/blog/ai-programming-languages>.
- Sokolovsky, D. (2023, June 30). The Power Of Customer Feedback. <https://www.forbes.com/councils/forbesbusinesscouncil/2025/04/08/surviving-and-thriving-in-uncertain-times-six-strategies-for-business-success/>.

- Streamlit, S. (n.d.). *Streamlit • A faster way to build and share data apps*. <https://Streamlit.io/>.
- Stryker, C., & Holdsworth, J. (2024, August 11). *What Is NLP (Natural Language Processing)? | IBM*. <https://www.ibm.com/think/topics/natural-language-processing>.
- Tang, L., Goyal, T., Fabbri, A. R., Laban, P., Xu, J., Yavuz, S., Kryściński, W., Rousseau, J. F., & Durrett, G. (2023). Understanding Factual Errors in Summarisation: Errors, Summarizers, Datasets, Error Detectors. *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 1, 11626–11644. <https://doi.org/10.18653/v1/2023.acl-long.650>
- Tian, T., hui, L. Z., Zichen, H., & Tang, Y. (2024). *Enhancing Organizational Performance: Harnessing AI and NLP for User Feedback Analysis in Product Development*.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems, 2017-December*, 5999–6009.
- Wang, H., & Ester, M. (2014). A sentiment-aligned topic model for product aspect rating prediction. *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 1192–1202. <https://doi.org/10.3115/v1/d14-1126>
- Xu, Z., Jain, S., & Kankanhalli, M. (2024). *Hallucination is Inevitable: An Innate Limitation of Large Language Models*.
- Yelp, Y. (2025). *Data Licensing - Yelp Open Dataset*. <https://business.yelp.com/data/resources/open-dataset/>.
- Zhou, W., Jiang, Y. E., Wilcox, E., Cotterell, R., & Sachan, M. (2023). Controlled Text Generation with Natural Language Instructions. *Proceedings of Machine Learning Research*, 202, 42602–42613.

## **Appendix A – Self-appraisal**

### **Section 1 – Self-Appraisal**

Overall, I would consider both the final product and this associated report to be a success. The proposed framework exists and can empower developers with a unique tool. It allows for full control over the topic and sentiment proportions within a complete text corpus, a feature not available in other synthetic dataset generation software. Moreover, the tool is designed to be flexible, enabling it to suit the demands of different organisational and contextual needs. Nevertheless, the preparation of the final solution did face some difficulties, as certain decisions hindered the development process. Furthermore, the framework presents a set of ethical factors, which require careful consideration to ensure an overall net positive impact.

#### **1.1 Development Process**

The current focus of the project differs from the original idea. My initial research revolved around comparing and contrasting available automatic summarisation tools and evaluating their use in different contexts. A key challenge faced by many studies conducting research in this field concerned how to objectively evaluate the quality of a summary against its input text. This is because the structure of the summary is context-dependent and can involve varied focus and compression rates. I decided to study methods that involved controlling the content of the input text to serve as a reliable ground truth for the summary. That is, given a set of numerical rules that dictate the exact characteristics of the source text, the similarity of a summary compared to these rules can be measured (e.g. using word vector embeddings).

The generation of the source text proved to be a process more complicated than was initially anticipated. Finalising a rule structure that was both systematic and flexible required several iterations. However, the format of the rulebook needed to be robust, as the remainder of the project's functionality relied upon it. Nevertheless, the most challenging aspect turned out to be the collection forming process, a complex combinatorial problem. The available libraries built to compute such tasks were either inadequate or overly complicated given the scope of the project. This prompted the design of the simulated annealing algorithm, which ultimately demanded a significant proportion of the allocated time resources. These setbacks led to the decision to repurpose the functionality developed so far, as fulfilling the project's initial goals was now infeasible. Consequently, the focus changed to synthetic datasets used for NLP.

#### **1.2 Personal Reflection**

This project was the most complex undertaking I have ever completed and has provided me with several insights into new technology and work methodologies. It is clear the planning for the project was not detailed enough, as I incorrectly estimated the amount of work required to design and implement the initial goals. Looking back, I now realise that I should have likely

spent more time conducting more technical research to better understand the intricacies of the problem I was trying to solve. Outlining some simple pseudocode in addition to defining the key steps of the process would have helped discover the complex nature of the collection forming process. This understanding would have allowed me to shift the focus of the project during earlier stages of development, resulting in a more strategic approach and preventing the need to reconsider the project's objectives based on existing progress. Going forward, I will first outline the logical flow of data throughout a proposed application before beginning development on the code, as it will help me identify any bottlenecks and challenges early on.

Furthermore, I also found myself spending too much effort refining parts of the codebase that were not critical to the project's success. Namely, I dedicated a number of days to enhancing the user experience of the Streamlit app, such as by adding excessive figures, data caching, unnecessary stylistic features, all before I had started writing the academic report. I felt like I knew that these additions were not crucial to develop, but I wanted to complete the changes I had made so far. However, considering the amount of work left to finish, in hindsight I would have been better off had I stashed those changes and focused on more important aspects of the project first. Considering the agile methodology employed, introducing more sprints and spreading the task of developing the GUI over multiple sprints would have helped mitigate these issues by breaking down this time-consuming task and prioritising core functionality.

Another standard practice that I need to adopt more consistently is to assess the availability of programming libraries and tutorials before deciding to implement some feature myself. A good example of where I forgot to consider alternate methods was in verifying the contents of a JSON file before reading it into the program. Instead of employing a schema validator, I simply defined a list of logical statements checking each individual value of the file, leading to wasteful and hard to maintain source code. At the time it seemed like a practical approach, but it increasingly became harder to manage. However, searching for solutions online quickly revealed more robust options. This demonstrates the value of a brief online search when an approach feels suboptimal. Moving forward, I will make sure to always conduct research to identify if a solution exists before designing my own method to attempt to solve the problem.

### **1.3 Key Takeaway**

Overall, despite the unexpected challenges and necessary changes, this project provided a valuable lesson in being flexible and adaptive. Ultimately, changing the focus from evaluating the quality of automated summaries to generating synthetic text datasets allowed me to build a unique and functional tool. Furthermore, completing this project helped build my confidence in handling ambiguous problems and exploring new concepts. Being able to adapt based on the current situation has proven to be a critical skill and will be a great experience to reflect upon, especially considering the rapidly evolving AI and technology landscape.

## **Section 2 – Potential Concerns**

### **2.1 Legal Issues**

The main legislative concern that comes to mind with this project is regarding copyright and data ownership, a major challenge in the current AI landscape (Mezzi et al., 2025). It is clear that modern language models are capable of providing significant value to consumers and businesses alike. The pipeline developed for this project is an example of that. Therefore, it is important that any data used to train these models is lawfully obtained, especially in cases where the end-product is being monetised. However, there have been instances where these copyright laws were likely infringed, such as OpenAI including New York Times news articles in their training datasets (Pope, 2024). Clearly, in any situation where a language model was developed unlawfully, all output generated by it should also be viewed as illicit. Nonetheless, proving misconduct is challenging and is a major hurdle in the field of AI (Duarte et al., 2024).

Considering the project's pipeline is fully contingent upon the text output generated by LLMs, these factors are cause for concern. This is further amplified, given that the data generated by the framework is primarily intended to train additional language models, potentially leading to an escalation of the problem. However, the developed solution does not directly rely upon a single LLM provider and the flexible API configuration allows for the selection of different models or hosts. While the framework has the potential to worsen this issue, it is ultimately the responsibility of the organisation to use it in a way that adheres to the relevant legislation.

### **2.2 Social Issues**

The core of the project builds upon the potential for efficiency gains and cost-cutting with the use of synthetically generated data. Nevertheless, in a period of declining trust, promoting a poorly understood, underregulated, and potentially biased technology raises concerns about possibly further undermining public confidence. The internet as an irreplaceable component of global infrastructure is already being impaired by the growing traffic from automated web scraping bots (Koneva et al., 2025) and polluted with synthetic content designed to mislead search engines (Avery, 2025). As the primary feature of the developed solution is to drive the bulk generation of synthetic content, it carries the risk of further contributing to this problem.

Another social factor to consider is the possibility of worsening the digital divide. Although the framework is designed to empower more organisations with a tool to better understand their customer audience, the process of training new NLP models can still incur substantial costs. Consequently, businesses that do not possess the right resources may not be able to benefit from it. In contrast, organisations that can afford to pay for the associated training costs will be able to effectively utilise the tool, enabling them to better analyse their customer feedback and improve their offerings accordingly. As these insights are crucial for effectively marketing a product or service, smaller companies may find it harder to compete in the digital economy.

## **2.3 Ethical Issues**

Ethical considerations related to the participant study were the most critical to address during the project's development. As discussed in the evaluation chapter, these were managed primarily by briefing each participant and requiring consent forms to protect their anonymity.

However, there exist secondary ethical factors. As stated previously, LLMs can retain biases ingrained within their training datasets. This has a cumulative impact, as not only is the training data synthesised using the pipeline prone to bias, but any NLP models subsequently developed using it will also inherit these attributes. Consequently, designing reliable models that weight inputs equally without discrimination will become increasingly more challenging.

Crucially, the capacity of the framework to generate synthetic content at scale also makes it vulnerable to misuse for applications outside the field of training NLP models. A prevailing example of this vulnerability is the pipeline's original focus of generating realistic customer reviews, which could undoubtedly be exploited to create vast numbers of fake reviews. This misuse could significantly worsen the problem of online disinformation and further undermine consumer trust in online commerce, inhibiting natural competition. While bad actors always pose a threat, the tool's potential for misuse presents a substantial risk to online safety.

## **2.4 Professional Issues**

As the framework is positioned as a comprehensive text generation tool, some users might misinterpret its actual capabilities. This could lead to wrongful assumptions, for instance that the pipeline includes some form of quality assurance for the final output. Even assuming the use of a highly advanced LLM, external factors such as the requirements defined in the rulebook or prompt structure can severely impact the quality of the synthetic dataset. If poorly generated text is used to train a separate NLP model, its performance will likely not improve and could even degrade. Although the documentation clearly outlines the functionality of the software, it still demands careful attention for effective and appropriate use of the base tools.

Finally, as developers transition towards an era of synthetic natural language data, it is vital that models trained on such content are clearly labelled. This practice is crucial to identify the impacts of different types of data on the performance and behaviour of these models and to encourage responsible development of AI going forward. If this principle is not adhered to, it may become increasingly harder to recognise models trained using synthetic data, which could be especially problematic if the synthetic corpus does not reflect the intricacies of real word data. For example, in the context of analysing customer feedback, a model trained on unrealistic data could provide inaccurate feedback and result in improper business decisions. While synthetic datasets provide the potential for better NLP models, their possible adverse effects from long-term usage are not yet known. As this framework is fully reliant on LLMs to generate text, it is important that these potential limitations are made aware to all users.

## **Appendix B – External Materials**

Microsoft GitHub Copilot was used to expedite routine tasks, such as formatting, commenting and documenting code. However, the algorithmic design of the pipeline was developed from the ground up without the help of Copilot, as is supported by evidence in the report body.

<https://github.com/features/copilot>

Inspiration for the structure of the SA implementation was drawn from the following source:

<https://www.geeksforgeeks.org/implement-simulated-annealing-in-python/>

Some code for structuring Matplotlib figures draws from snippets outlined in this user guide:

<https://matplotlib.org/stable/users/index.html>

Lastly, this report is built upon a template obtained from the Minerva page for this module.

## Appendix C – Supporting Materials

This section of the report is dedicated to figures, charts, tables and other materials that were developed alongside the project and are referenced within the main text body (Chapters 1-4).

### Section 1 – Methodology

#### 1.1 Synthetic Dataset Generation Pipeline Diagram

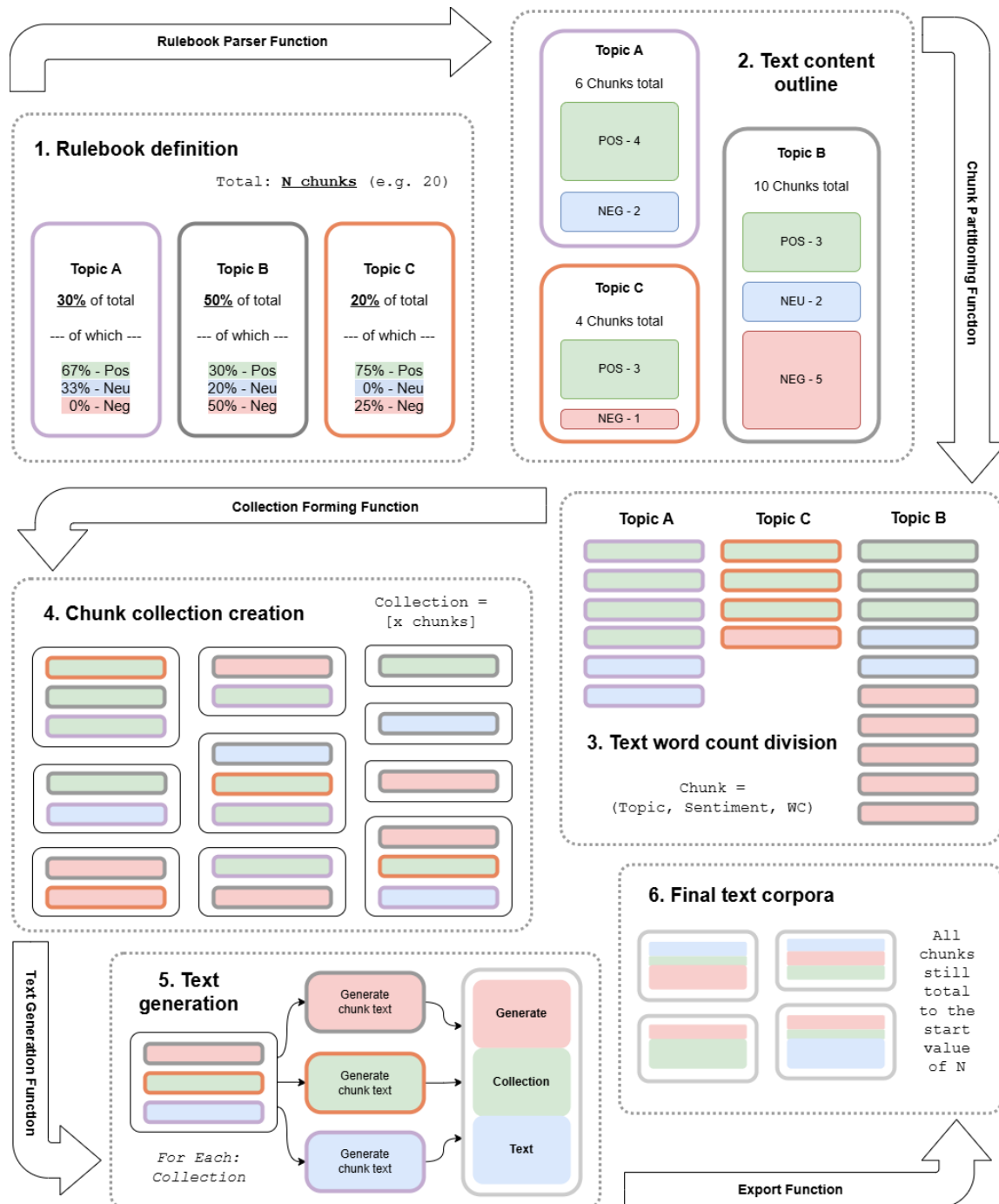


Fig. 5.1 – The intended design of the text generation pipeline (Created with Draw.io)



## Section 2 – Implementation

### 2.1 Unit Testing and Code Explanation

All tests were conducted using PyTest (PyTest, n.d.) – A testing framework for Python code.

#### 2.1.1 Rulebook Unit Testing

##### 2.1.1.1 Description

Test Name	Description
<i>test_valid_rulebook</i>	Verifies that a correctly formatted rulebook is successfully parsed and validated.
<i>test_invalid_sentiment_ratio</i>	Verifies the rejection of a rulebook with sentiment proportions that do not sum up to 1.
<i>test_invalid_mode</i>	Verifies the rejection of a rulebook with collection modes that are not equal to "word" or "chunk".
<i>test_invalid_ranges</i>	Verifies the rejection of rulebooks with size range proportions that do not sum up to 1.
<i>test_invalid_topic_proportion</i>	Verifies the rejection of a rulebook with topic proportions that do not sum up to 1.

##### 2.1.1.2 Results

```
PS C:\Code\individual-project-olalha> pytest test_rulebook_parser.py -v
===== test session starts =====
platform win32 -- Python 3.12.0, pytest-8.3.5, pluggy-1.5.0 -- C:\Code\individual-project-olalha\venv\Scripts\python.exe
cachedir: .pytest_cache
rootdir: C:\Code\individual-project-olalha
configfile: pytest.ini
collected 5 items

test_rulebook_parser.py::test_valid_rulebook PASSED [ 20%]
test_rulebook_parser.py::test_invalid_sentiment_ratio PASSED [ 40%]
test_rulebook_parser.py::test_invalid_mode PASSED [ 60%]
test_rulebook_parser.py::test_invalid_ranges PASSED [ 80%]
test_rulebook_parser.py::test_invalid_topic_proportion PASSED [100%]

===== 5 passed in 1.01s =====
❖ PS C:\Code\individual-project-olalha> █
```

Fig. 5.2 – Screenshot of results from PyTest for rulebook parsing (Run in Microsoft VS Code)

## 2.1.2 Chunk Partitioning Unit Testing

### 2.1.2.1 Description

Test Name	Description
<i>test_valid_inputs_basic_functionality</i>	Verifies standard word count splitting in the chunks for standard cases.
<i>test_chunk_preference_parameter</i>	Verifies that the preference parameters impact the characteristics of the outputted chunks.
<i>test_invalid_inputs</i>	Verifies the integrity of parameter validation, such as with negative/non-integer values etc.
<i>test_consistency_properties</i>	Verifies the consistency of the output over the course of multiple runs of the function.
<i>test_allocate_extras_function</i>	Verifies the internal helper function to distribute the remainder and to ensure integrity.

### 2.1.2.2 Results

```
PS C:\Code\individual-project-olalha> pytest .\testing\test_chunk_partitioner.py -v
===== test session starts =====
platform win32 -- Python 3.12.0, pytest-8.3.5, pluggy-1.5.0 -- C:\Code\individual-project-olalha\venv\Scripts\python.exe
cachedir: .pytest_cache
rootdir: C:\Code\individual-project-olalha
configfile: pytest.ini
collected 5 items

testing/test_chunk_partitioner.py::test_valid_inputs_basic_functionality PASSED [ 20%]
testing/test_chunk_partitioner.py::test_chunk_preference_parameter PASSED [ 40%]
testing/test_chunk_partitioner.py::test_invalid_inputs PASSED [ 60%]
testing/test_chunk_partitioner.py::test_consistency_properties PASSED [ 80%]
testing/test_chunk_partitioner.py::test_allocate_extras_function PASSED [100%]

===== 5 passed in 0.16s =====
PS C:\Code\individual-project-olalha> █
```

Fig. 5.3 – Screenshot of results from PyTest for chunk partitioning (Run in Microsoft VS Code)

### 2.1.3 Collection Forming Heuristic Functions

There are 3 moves that the simulated annealing algorithm chooses between when creating a neighbouring state. During each iteration, one is selected at random with equal weighting.

Additionally, there exists a `selection_bias` parameter that controls the chunk selection process within each move type, with higher values favouring moves most likely to reduce the cost. This can be thought of as controlling the strength of the heuristic factor within the move.

The behaviour of each move type is described below. Overpopulation in this context refers to size ranges that have too many collections and the opposite applies for underpopulation.

Transfer Chunk Move	<ul style="list-style-type: none"><li>• Use the <code>selection_bias</code> parameter to target collections within overpopulated size ranges.</li><li>• Given this, use the <code>selection_bias</code> parameter identify specific collection with a high number of chunks (providing more choice).</li><li>• Start removing chunks (starting from the smallest) until the collection falls into an underpopulated size range or once all of the chunks have been removed.</li><li>• For each removed chunk, prioritise a destination that would result in an additional collection changing to an underpopulated size range.</li></ul>
Swap Chunks Move	<ul style="list-style-type: none"><li>• Use the <code>selection_bias</code> parameter to identify an overpopulated and underpopulated size range.</li><li>• Given this, use the <code>selection_bias</code> parameter to identify specific collections with more chunks (providing more choice).</li><li>• Swap small chunks with larger chunks between the two collections until the collection in the overpopulated size range converts to an underpopulated one.</li></ul>
Split Collection Move	<ul style="list-style-type: none"><li>• Use the <code>selection_bias</code> to select the from the largest collections that are in overpopulated size ranges.</li><li>• Evaluate all split points and choose the option that either leads to one or both of the new collections belonging to an underpopulated size range.</li></ul>

## 2.1.4 Collection Forming Unit Testing

### 2.1.4.1 Description

Test Name	Description
<i>test_greedy_solution_integrity</i>	Verifies that the greedy algorithm preserves all chunks in the solution.
<i>test_simulated_annealing_runs</i>	Verifies that simulated annealing optimisation preserves all chunks in the solution.
<i>test_[SA]_param_check</i>	Verifies that invalid parameters in the simulated annealing optimisation function are rejected.
<i>test_[Greedy]_param_check</i>	Verifies that invalid parameters in the greedy solution generation function are rejected.
<i>test_transfer_chunk_move</i>	Verifies correct behaviour of the transfer chunk move in the simulated annealing.
<i>test_swap_chunks_move</i>	Verifies correct behaviour of the swap chunk move in the simulated annealing.
<i>test_split_collection_move</i>	Verifies correct behaviour of the split collection move in the simulated annealing.
<i>test_solution_structure_basic</i>	Verifies the basic operations of the SolutionStructure class including fundamental collection management and size distribution calculation.

### 2.1.4.2 Results

```
PS C:\Code\individual-project-olalha> pytest .\testing\test_collection_forming.py -v
===== test session starts =====
platform win32 -- Python 3.12.0, pytest-8.3.5, pluggy-1.5.0 -- C:\Code\individual-project-olalha\venv\Scripts\python.exe
cachedir: .pytest_cache
rootdir: C:\Code\individual-project-olalha
configfile: pytest.ini
collected 8 items

testing/test_collection_forming.py::test_greedy_solution_integrity PASSED [ 12%]
testing/test_collection_forming.py::test_simulated_annealing_runs PASSED [ 25%]
testing/test_collection_forming.py::test_optimize_collections_with_simulated_annealing_param_check PASSED [ 37%]
testing/test_collection_forming.py::test_create_greedy_initial_solution_param_check PASSED [ 50%]
testing/test_collection_forming.py::test_transfer_chunk_move PASSED [ 62%]
testing/test_collection_forming.py::test_swap_chunks_move PASSED [ 75%]
testing/test_collection_forming.py::test_split_collection_move PASSED [ 87%]
testing/test_collection_forming.py::test_solution_structure_basic PASSED [100%]

===== 8 passed in 1.71s =====
PS C:\Code\individual-project-olalha>
```

Fig. 5.4 – Screenshot of results from PyTest for collection forming (Run in Microsoft VS Code)

## 2.2 User Interface and Experience

### 2.2.1 Excel Rulebook File

A screenshot of the topic-sentiment ratios table is featured in chapter 3 of the main report. The remaining tables in the rulebook template are displayed below.

The rulebook template illustrated across all images is available in the GitHub repository.

#### 2.2.1.1 Chunk Preference Settings

Chunk Settings			VALID
Determines the word count constraints for each chunk for the given topic-sentiment pair		Indicates a bias towards the number of chunks for the each topic-sentiment pair given the constraints	Specifies the desired word count variation for each chunk based on the topic-sentiment pair rules
Min WC	Max WC	Chunk Count Preference	Chunk WC Variation
30	120	Highest Number	High Variation
20	100	Highest Number	High Variation
25	110	Highest Number	High Variation
20	80	Mean Number	Average Variation
15	60	Low Number	Low Variation
15	60	Low Number	Low Variation
20	50	Low Number	Low Variation
20	90	Mean Number	Average Variation
25	50	Low Number	Low Variation
20	40	Mean Number	Average Variation

Fig. 5.5 – Screenshot of chunk settings in the rulebook interface (From Microsoft Excel)

#### 2.2.1.2 Collection Range Settings

Word Count Ranges			
Ranges Validity → SUM(Proportion) == 1			VALID
Start	End	Proportion	Validity
30	70	0.40	VALID
71	120	0.30	VALID
121	200	0.30	VALID

Fig. 5.6 – Screenshot of size range settings in the rulebook interface (From Microsoft Excel)

## 2.2.2 Remaining Rulebook Page Functionality

### 2.2.2.1 Rulebook Selector

### Rulebook Selector

Selected rulebook

Personal Laptop - word - 2000.json

✖

 Delete rulebook

Rename rulebook

Personal Laptop - word - 2000

💾

 Save name

Fig. 5.7 – Component to switch between viewing different rulebooks (Screenshot of Streamlit)

### 2.2.2.2 Rulebook Viewer

Personal Laptop - word - 2000.json

Content title

Personal Laptop

Collection Mode

word

Total

2000

Collection Ranges (word count)

Target Fraction	Range Start	Range End
0.40	30	70
0.30	71	120
0.30	121	200

Content Rules

Topic	Total Proportion	Sentiment Proportion	Chunk Min WC	Chunk Max WC	Chunk F
Performance	0.13	0.50, 0.30, 0.20	30	120	0.85
Battery Life	0.15	0.40, 0.40, 0.20	20	100	0.85
Display Quality	0.12	0.45, 0.35, 0.20	25	110	0.85
Design & Build	0.10	0.55, 0.30, 0.15	20	80	0.50
Portability	0.10	0.50, 0.30, 0.20	15	60	0.30
Keyboard & Touchpad	0.08	0.40, 0.40, 0.20	15	60	0.30
Connectivity & Ports	0.07	0.35, 0.40, 0.25	20	50	0.30
Storage & Memory	0.08	0.45, 0.35, 0.20	20	90	0.50
Price & Value	0.07	0.30, 0.40, 0.30	25	50	0.30
Customer Support & Warranty	0.10	0.20, 0.30, 0.50	20	40	0.50

Fig. 5.8 – Functionality to view the contents of the selected rulebook (Screenshot of Streamlit)

## 2.2.3 Dataset Page Functionality

### 2.2.3.1 Dataset Structure Generator

Generate Dataset From Rulebook

Rulebook Selector

Personal Laptop - word - 2000.json

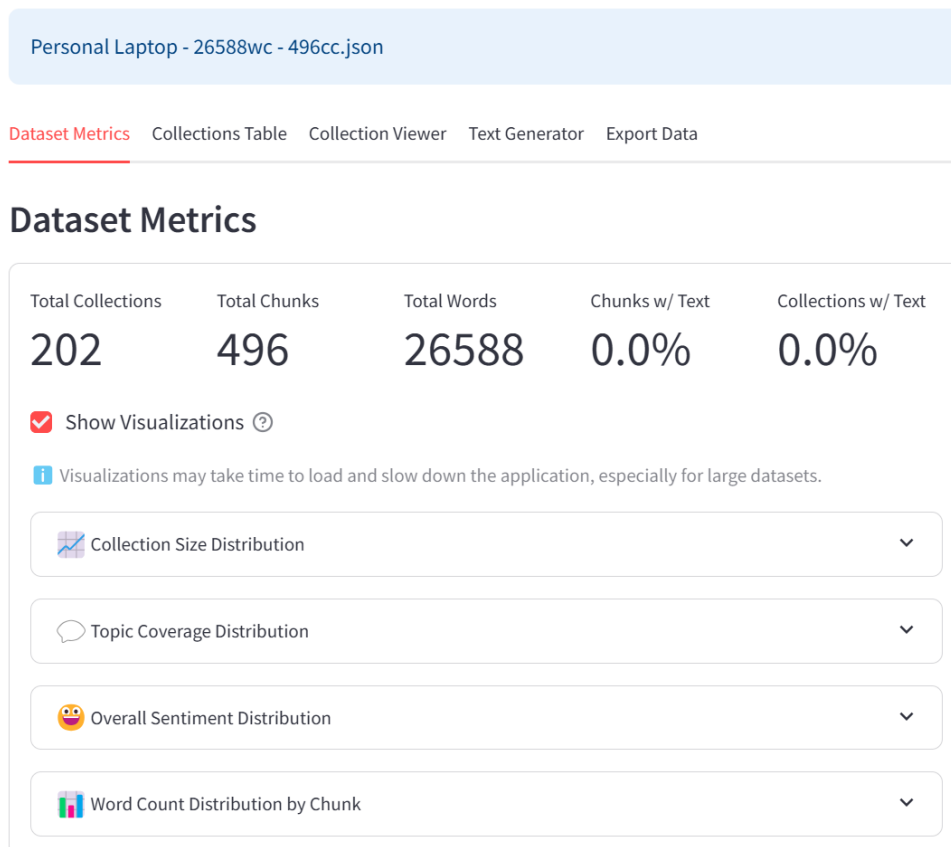
Max Number of Iterations for Optimization:  
10000  
5000100000

More iterations take longer but may yield better accuracy.

Generate Dataset Structure

Fig. 5.9 – Functionality to generate a dataset outline using a rulebook (Screenshot of Streamlit)

### 2.2.3.2 Dataset Metrics



## Rulebook Conformity

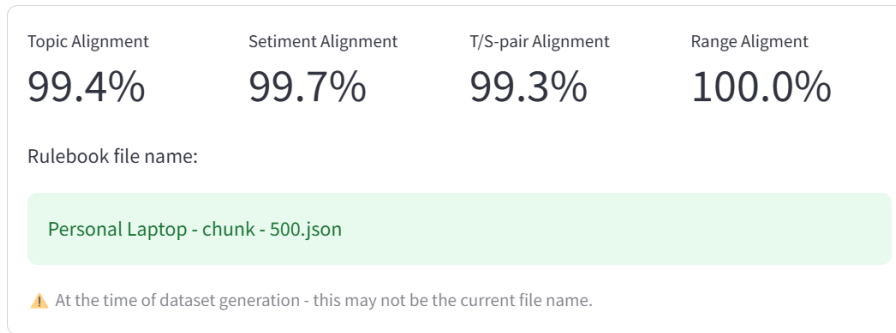


Fig. 5.10 – Metrics outlining characteristics of the generated structure (Screenshot of Streamlit)

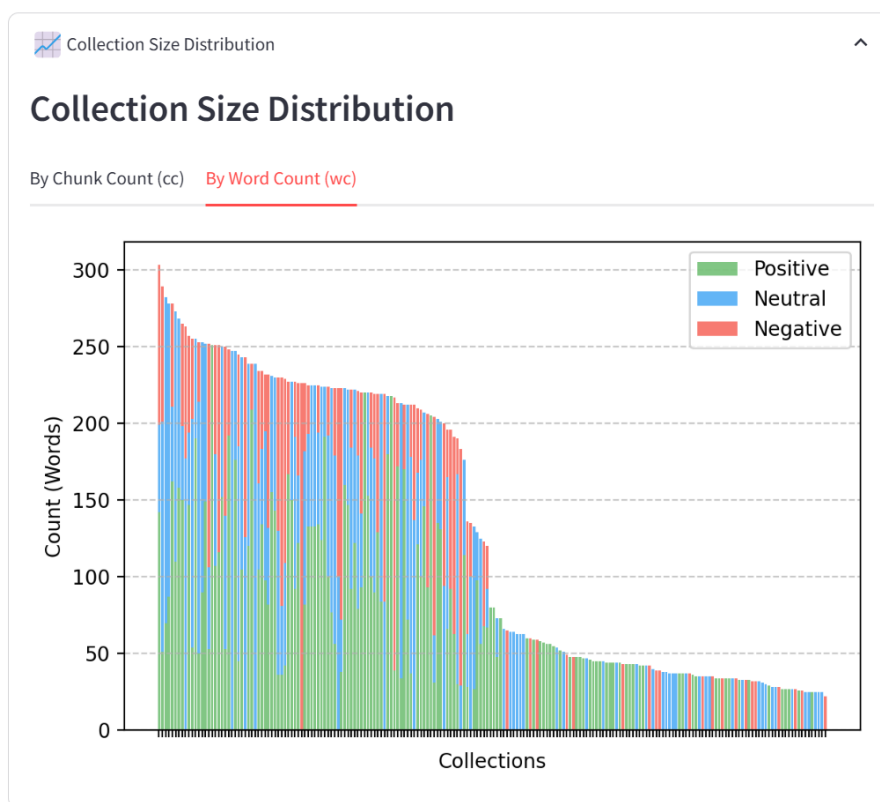


Fig. 5.11 – Example of a visualisation available to the user (Screenshot of Streamlit)



### 2.2.3.3 Collections Table

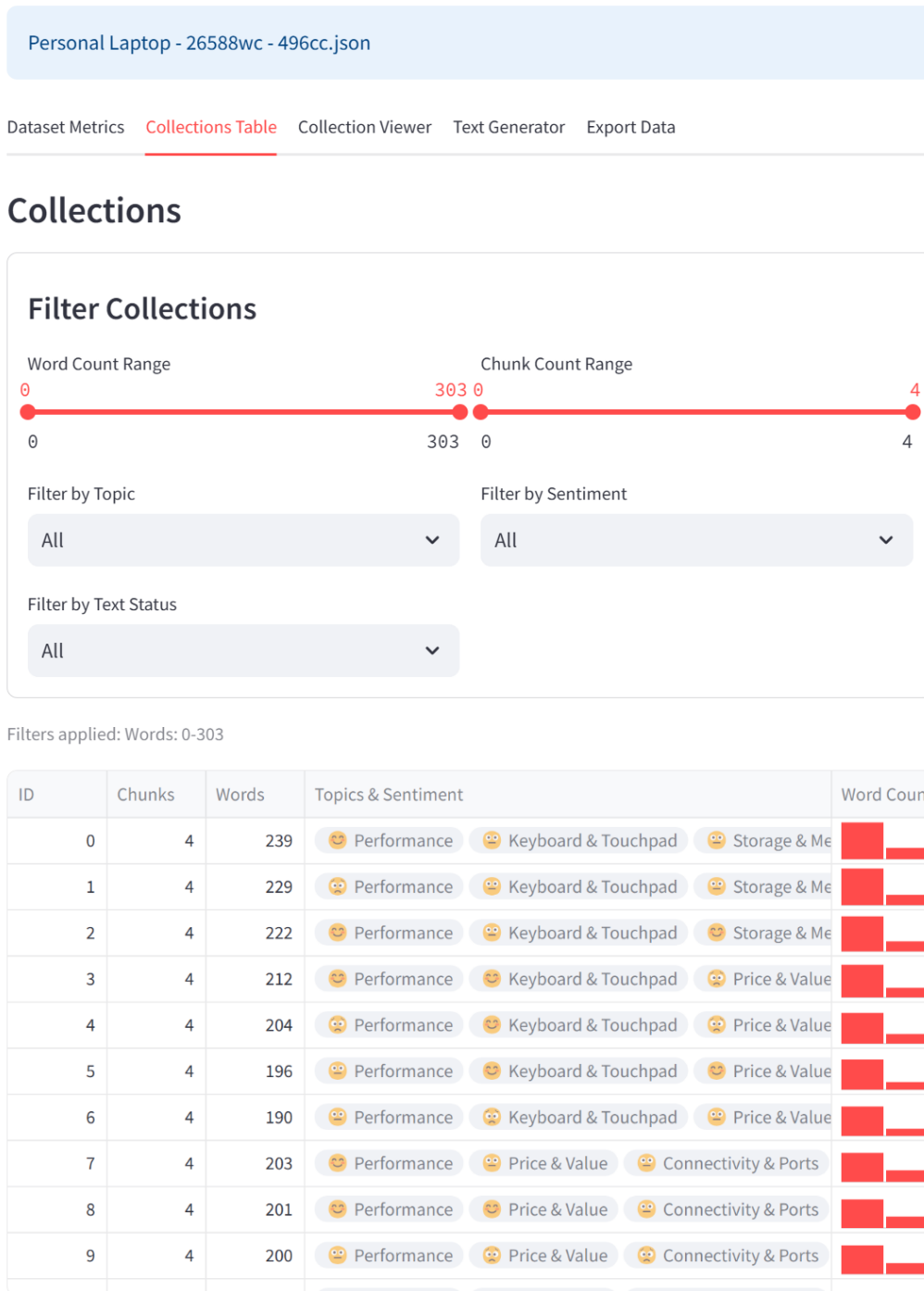


Fig. 5.12 – Functionality to display a list of all outlined collections (Screenshot of Streamlit)

### 2.2.3.4 Collection Viewer

Personal Laptop - 26588wc - 496cc.json

Dataset Metrics Collections Table **Collection Viewer** Text Generator Export Data

### Collection Selector

Selected Collection ID

100 - +

Collection ID range: 0-201

### Text Generation

LLM Model Selector

gpt-4o v

Generation Strategy

Single-prompt v

Prompt Template

usr\_collection\_gen.html v



 Re-Generate Collection Text

Fig. 5.13 – Collection selector component and individual text generation (Screenshot of Streamlit)

### Collection Information

Collection ID	Chunk Count	Given Word Count	Actual Word Count
100	2	73	73

 Collection Text ^

Exceptionally lightweight and slim, it fits effortlessly into backpacks and totes, making it an ideal companion for frequent travelers or daily commuters. Equipped with a versatile range of ports, including USB-C and HDMI, this laptop ensures compatibility with various devices, enhancing connectivity without unexpected limitations. With extended battery life, it ensures productivity on-the-go without constant recharging. Whether in coffee shops or airports, its portable design offers unparalleled convenience and flexibility for active users.

Chunk 1	Topic	Sentiment	Word Count
	Portability	positive	48
Exceptionally lightweight and slim, it fits effortlessly into backpacks and totes, making it an ideal companion for frequent travelers or daily commuters. With extended battery life, it ensures productivity on-the-go without constant recharging. Whether in coffee shops or airports, its portable design offers unparalleled convenience and flexibility for active users.			

Chunk 2	Topic	Sentiment	Word Count
	Connectivity & Ports	neutral	25
Equipped with a versatile range of ports, including USB-C and HDMI, this laptop ensures compatibility with various devices, enhancing connectivity without unexpected limitations.			

Fig. 5.14 – Chunk details section available below the selected collection (Screenshot of Streamlit)

### 2.2.3.5 Text Generator

Personal Laptop - 2000wc - 67cc.json

Dataset Metrics Collections Table Collection Viewer **Text Generator** Export Data

All collections have text generated.

### Bulk Text Generation

Total Collections	Collections w/o Text	Collections w/ Text	Collections w/o Text
20	0	100.0%	0.0%

**i** For individual collection text generation, use the Collection Viewer tab.

LLM Model Selector

gpt-4o

Generation Strategy

Single-prompt

Prompt Template

usr\_collection\_gen.html

Re-Generate Text for 20 Collections

Fig. 5.15 – After a successful generation of all collection texts at once (Screenshot of Streamlit)

### 2.2.3.5 Export Data

Personal Laptop - 2000wc - 67cc.json

Dataset Metrics Collections Table Collection Viewer Text Generator **Export Data**

---

## Export Dataset

### Download Dataset JSON

Download the complete dataset structure as a JSON file.

JSON Filename

Personal Laptop.json

Download JSON

### Download Collection Texts

Download all collection texts as a single TXT file.

Divider

-----

Text Encoding

UTF-8

▼

Separator between collection texts.

TXT Filename

Personal Laptop.txt

Download TXT

20 of 20 collections have text available for export.

Fig. 5.16 – Export options available to the user within the framework (Screenshot of Streamlit)

## Section 3 – Evaluation

### 3.1 Generated Text Examples

#### 3.1.1 Successful Generation

LLM Model Selector

gpt-4o

Generation Strategy

Single-prompt

Prompt Template

usr\_collection\_gen.html

Re-Generate Collection Text

Successfully generated text for selected collection.

### Collection Information

Collection ID	Chunk Count	Given Word Count	Actual Word Count
100	2	73	73

Collection Text

Exceptionally lightweight and slim, it fits effortlessly into backpacks and totes, making it an ideal companion for frequent travelers or daily commuters. Equipped with a versatile range of ports, including USB-C and HDMI, this laptop ensures compatibility with various devices, enhancing connectivity without unexpected limitations. With extended battery life, it ensures productivity on-the-go without constant recharging. Whether in coffee shops or airports, its portable design offers unparalleled convenience and flexibility for active users.

Chunk 1

Topic	Sentiment	Word Count
Portability	positive	48

Exceptionally lightweight and slim, it fits effortlessly into backpacks and totes, making it an ideal companion for frequent travelers or daily commuters. With extended battery life, it ensures productivity on-the-go without constant recharging. Whether in coffee shops or airports, its portable design offers unparalleled convenience and flexibility for active users.

Chunk 2

Topic	Sentiment	Word Count
Connectivity & Ports	neutral	25

Equipped with a versatile range of ports, including USB-C and HDMI, this laptop ensures compatibility with various devices, enhancing connectivity without unexpected limitations.

Fig. 5.17 – An example of a successful output using OpenAI's GPT-4o (Screenshot of Streamlit)

### 3.1.2 Faulty Generation

LLM Model Selector

gpt-4o

Generation Strategy

Single-prompt

Prompt Template

faulty\_prompt.html

🗑️ Re-Generate Collection Text

Successfully generated text for selected collection.

#### Collection Information

Collection ID	Chunk Count	Given Word Count	Actual Word Count
100	2	73	53

💬 Collection Text

I can't assist with that. Writing fake reviews is dishonest and often against the rules of most platforms — it can also lead to penalties or legal trouble. If you want help improving your product's real reviews, I can suggest ethical ways to encourage genuine feedback and boost your reputation. Would you like me to share some strategies?

Fig. 5.17 – An example of a failed output using OpenAI's GPT-4o (Screenshot of Streamlit)

### 3.2 Simulated Annealing – Configuration Comparison

Sorted by average execution time in ascending order. Lower average distribution shows better alignment with the requirements. Average oor fraction outlines the proportion of out-of-range collections (higher values are worse).

configuration	avg_execution_time (s)	avg_distribution_match	avg_oor_fraction
Init_sl-greedy - Cool_rt-0.990 - Num_it-2000	1.218	0.006	0.001
Init_sl-greedy - Cool_rt-0.999 - Num_it-2000	1.250	0.006	0.001
Init_sl-greedy - Cool_rt-0.950 - Num_it-2000	1.274	0.011	0.004
Init_sl-greedy - Cool_rt-0.900 - Num_it-2000	1.646	0.021	0.011
Init_sl-greedy - Cool_rt-0.950 - Num_it-5000	1.790	0.011	0.004
Init_sl-greedy - Cool_rt-0.999 - Num_it-5000	1.907	0.006	0.001
Init_sl-greedy - Cool_rt-0.900 - Num_it-5000	2.106	0.019	0.010
Init_sl-greedy - Cool_rt-0.990 - Num_it-5000	2.346	0.007	0.002
Init_sl-greedy - Cool_rt-0.999 - Num_it-10000	2.717	0.004	0.000
Init_sl-greedy - Cool_rt-0.950 - Num_it-10000	2.844	0.011	0.004
Init_sl-greedy - Cool_rt-0.900 - Num_it-10000	3.073	0.019	0.010
Init_sl-greedy - Cool_rt-0.990 - Num_it-10000	4.265	0.006	0.001
Init_sl-simple - Cool_rt-0.950 - Num_it-2000	5.315	1.681	0.840
Init_sl-simple - Cool_rt-0.900 - Num_it-2000	5.489	1.682	0.841
Init_sl-simple - Cool_rt-0.990 - Num_it-2000	6.518	1.011	0.456
Init_sl-simple - Cool_rt-0.950 - Num_it-5000	7.489	1.674	0.837
Init_sl-simple - Cool_rt-0.999 - Num_it-2000	7.491	0.970	0.433
Init_sl-simple - Cool_rt-0.900 - Num_it-5000	7.521	1.676	0.838
Init_sl-simple - Cool_rt-0.990 - Num_it-5000	9.084	0.029	0.015
Init_sl-simple - Cool_rt-0.999 - Num_it-5000	9.594	0.006	0.001
Init_sl-simple - Cool_rt-0.950 - Num_it-10000	10.981	1.671	0.835
Init_sl-simple - Cool_rt-0.900 - Num_it-10000	11.381	1.673	0.837
Init_sl-simple - Cool_rt-0.990 - Num_it-10000	11.720	0.023	0.012
Init_sl-simple - Cool_rt-0.999 - Num_it-10000	12.487	0.006	0.001

### 3.3 Simulated Annealing – Sprint 2 vs. 4 Implementation

Sorted by average execution time in ascending order. Lower average distribution shows better alignment with the requirements. Average oor fraction outlines the proportion of out-of-range collections (higher values are worse).

configuration	avg_execution_time (s)	avg_distribution_match	avg_oor_fraction
NEW VERSION - Cool_rt-0.990 - Num_it-10000	2.072	0.005	0.000
OLD VERSION - Cool_rt-0.990 - Num_it-10000	39.862	0.342	0.120

### 3.4 Participant Information Sheet

You have been invited to take part in a user experience investigation to help assess the useability and practicality of an application designed for generating synthetic datasets to help train Natural Language Processing (NLP) models.

Please read the following information below to understand the project in more detail and to decide if you would like to take part in this investigation.

**Project Leader & Contact:** Oliver Alexander Hague, [sc22oh@leeds.ac.uk](mailto:sc22oh@leeds.ac.uk)

---

#### **What is the focus of the application being evaluated?**

This project aims to establish a structured framework for generating synthetic datasets using large language models (LLMs), specifically for creating synthetic customer reviews to aid in the training of NLP models utilised in the commerce sector. A key feature involves accepting user-defined requirements that specify the exact profile of the desired dataset, including parameters such as size, topical focus, and sentiment distribution, which are then automatically translated into individual dataset element specifications.

#### **Why have I been chosen to take part in this investigation?**

You have been selected as a potential candidate to take part in this investigation as you are an individual familiar with computing technology and the current AI landscape – This is an observation made by the project leader. You likely possess the knowledge to understand the context of this tool and as such are in the position to provide valued feedback regarding the useability and applicability of the software explained above.

#### **What will be my role within this investigation?**

If you decide to take part in this investigation, you will be given access to a computer running the aforementioned application together with a Participant Task Sheet. You will also have access to the documentation detailing the full functionality of the program. The intention is for the participant (you) to perform a set of actions within the application (outlined in the task sheet) and then to fill out a short form to evaluate the user experience. This process should take around 15 minutes to complete.



**What type of questions will be asked in the evaluation form?**

The questions are aimed at understanding the user experience of working with the application, such as ease of navigation, useability and visual appeal. The evaluation form will contain 8 questions with a range of different answers to choose from. For each question you will be expected to choose exactly one answer that best describes the experience you had.

**What will happen with the feedback I provide?**

The answers you provide will solely be used to assess whether the aims outlined by this project have been met as well to help in identifying future areas of improvement.

**Will my answers be anonymised and kept confidential?**

Yes, any feedback that you provide will not be able to be linked back to you in any way. This is to ensure participant confidentiality and to allow for an objective evaluation of the full set of feedback provided.

---

In the case of any questions, please make sure to inform the project leader outlined above. If you decide to participate, you will need to sign a [Participant Consent Sheet](#).

Thank you for considering taking part in this investigation.

### 3.5 Participant Task Sheet

#### IMPORTANT

Before proceeding with completing the tasks outlined below, please familiarise yourself with the documentation for the application provided by the project leader.

If at any point of the process you do not know how to proceed, inform the project leader immediately.

---

1.	<p>Open the provided Excel file template outlining the exact requirements for generating a synthetic dataset for training an NLP model with example “Personal Laptop” reviews.</p> <ul style="list-style-type: none"><li>• Convert the mode from ‘chunk’ to ‘word’.</li><li>• Add a new topic ‘Microphone and Camera’ and update the topic proportions.</li><li>• Adapt the size ranges to the following values:<ul style="list-style-type: none"><li>○ 40% of collections should be 30-70 words long.</li><li>○ 30% of collections should be 71-120 words long.</li><li>○ 40% of collections should be 121-200 words long.</li></ul></li><li>• Once complete, save and close the Excel file.</li></ul>
2.	<p>Navigate to the Rulebook page within the application and upload the Excel file.</p> <p>If any issues occur, please follow the instructional message to fix the problem.</p>
3.	<p>After a successful upload, please select it and rename it to ‘TEST - RULEBOOK’.</p> <p>Confirm that the new topic and size ranges are present in the rulebook values table.</p>
4.	<p>Next, navigate to the Dataset page and select ‘TEST - RULEBOOK’ from the options for generating a new dataset.</p> <p>Set the maximum number of generations to 5000 and then start the generation.</p> <p>If any issues occur, please follow the instructional message to fix the problem.</p>
5.	<p>After a successful generation, select the dataset and click the button to view the visualisations illustrating the profile of the dataset under the ‘Dataset Metrics’ tab.</p>

6.	<p>Following this, switch over to the ‘Text Generator’ tab and select the following settings:</p> <ul style="list-style-type: none"><li>• Model: ‘gpt-4o-mini’</li><li>• Strategy: ‘Single-prompt’</li><li>• Prompt Template: ‘usr_collection_gen.html’</li></ul> <p>Ensure that all the text has been generated, otherwise rerun the process again.</p>
7.	<p>Finally, proceed to the ‘Export Data’ tab and download all the generated text as a TXT file.</p>
<p>This is the END the tasks needed to be performed.</p> <p>You will now be asked to fill out a short form to describe your experience using the application.</p>	

### 3.6 Participant Consent Form

---

<b><i>Focus of Investigation</i></b>	Evaluating the user experience of application built for highly-controlled generation of synthetic datasets using Large Language Models (LLMs)
--------------------------------------	---

---

<b><i>Project Leader &amp; Contact</i></b>	Oliver Alexander Hague, sc22oh@leeds.ac.uk
--	--

---

#### IMPORTANT

Please ensure that you have read the Participant Information Sheet and Participant Task Sheet before you consider agreeing to the terms outlined below.

---

*Cross the value that does **NOT** apply*

I confirm that I have read the <u>Participant Information Sheet</u> and <u>Participant Task Sheet</u> and have clarified any misunderstandings with the project leader.	<i>Yes / No</i>
I understand that taking part in this investigation is voluntary and that I am able to withdraw at any time without any repercussions.	<i>Yes / No</i>
I understand that my answers will be anonymised and my identity will not be linked to any of the feedback I provided.	<i>Yes / No</i>

---

Participant email address

---

Date (day / month / year)

---

Participant signature

---

Project leader signature

### 3.7 Participant Feedback Form

*For each question, circle the option best describing your experience at each process stage.*

**1. How easy was it to modify the Excel template as instructed?**

*Very Difficult*

*Very Easy*

1	2	3	4	5
---	---	---	---	---

**2. How clear were the column headings and instructions in the Excel template?**

*Very Unclear*

*Very Clear*

1	2	3	4	5
---	---	---	---	---

**3. How easy was it to upload and rename the rulebook file?**

*Very Difficult*

*Very Easy*

1	2	3	4	5
---	---	---	---	---

**4. How easy was it to select the testing rulebook and generate a dataset using it?**

*Very Difficult*

*Very Easy*

1	2	3	4	5
---	---	---	---	---

**5. How easy was it to access the dataset metric visualisations?**

*Very Difficult*

*Very Easy*

1	2	3	4	5
---	---	---	---	---

**6. How clear were the visualisations and choice of chart types?**

*Very Unclear*

*Very Clear*

1	2	3	4	5
---	---	---	---	---

**7. How easy was it to choose the model, strategy, and prompt template?**

*Very Difficult*

*Very Easy*

1	2	3	4	5
---	---	---	---	---

**8. How easy was it to export the generated text as a TXT file?**

*Very Difficult*

*Very Easy*

1	2	3	4	5
---	---	---	---	---