Transformer-Based Embedding Models for System Similarity: A Comparative Study of RoBERTa and GPT

by

Isha Vipul Patel

(Under the Direction of Beshoy Morkos)

Abstract

The increasing adoption of language models in technical domains has raised curiosity about utilizing them to examine system design and specifications effectively. This research assesses how two popular LLMs, RoBERTa and GPT, can identify similarities among system technologies through their design and requirement details. Through a comparison of these model's performance, the study aims to gauge their capacity to categorize systems with traits and distinguish system attributes.

The study showed that GPT tends to make unique groups that can effectively capture subtle differences in system needs and details. This is in contrast to Roberta's tendency to combine system features into groups, which suggests that she has trouble accurately figuring out complex systems. Through methods like t-SNE projections and UMAP embedding, the research visually presents how each model excels in identifying patterns and distinguishing between system types.

The aim of this paper is to investigate how GPT and Roberta could evaluate systems in many different ways. GPT is adaptable enough to uncover variations between technological systems. The outcomes reveal how to choose models depending on the degree of complexity in the system design and requirement analysis.

Keywords: GPT, RoBERTa, Design, Embedding, Function, Form, Requirements, Descriptions, System, Similarity

Transformer-Based Embedding Models for System Similarity: A Comparative Study of RoBERTa and GPT

by

Isha Vipul Patel

MS I.T., Uka Tarsadia University, India, 2019

A Thesis Submitted to the Graduate Faculty of the

University of Georgia in Partial Fulfillment of the Requirements for the Degree.

Master of Science

Athens, Georgia

2024

Transformer-Based Embedding Models for System Similarity: A Comparative Study of RoBERTa and GPT

by

Isha Vipul Patel

Dedication

This work is dedicated to my father, whose belief in me never wavered and who inspired me greatly when I was unsure about pursuing a second master degree, especially in Artificial Intelligence. To my mother, who hates me leaving India to finish my degree in the United States but never stops pushing me to reach new heights; to my brother, who, despite his quiet path, never stops learning from my experiences; and to my friends, who listened to me patiently as I vented about life.

It is also dedicated to those loved ones who filled my life with laughter and touched my heart.

Thank you for standing by me and making this achievement possible.

Acknowledgments

I want to sincerely thank my family and friends for always standing by me and pushing me to be my best. Your support has meant the world to me. I'm especially grateful to Dr. Beshoy Morkos, my major professor, for guiding me every step of the way and offering advice and encouragement when I needed it most.

A big thank you to my thesis committee— Dr. Neal Outland and Dr. Fei dou —for believing in me, allowing me to explore my ideas, and steering me through this research even when I felt lost. I'm also thankful to the Institute of Artificial Intelligence at UGA, whose members helped me push through to the finish line. Lastly, I thank God for blessing me with the strength and opportunities to achieve this milestone.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As large language models like as GPT and RoBERTa gain popularity, their applications have moved beyond natural language processing to include system design and requirement analysis. Understanding the similarities and contrasts of system technologies has long been an important component of engineering, since these insights help make essential decisions throughout the product development process. Traditionally, system comparison has been a time-consuming, human activity; however, LLMs provide a chance to expedite this process through automated textual analysis. This study examines the usage of GPT and RoBERTa to capture system similarities and discriminate between complicated characteristics in system design and requirement specifications.

1.1 Motivation For The Study

The rising complexity of contemporary systems and the growing demand for instruments capable of efficiently analyzing and comparing them are the driving forces behind this research. Understanding how these systems relate to one another gets increasingly difficult when systems change and so do their design needs (Hein, Kames, et al. 2024; Htet Hein, Morkos, and Sen 2017; Morkos and Summers 2010; Morkos, Shankar, and Summers 2012). Engineers, product developers, and other stakeholders want tools that can quickly and accurately assess system similarities and variances in order to make educated decisions. LLMs have proven their worth in a variety of domains, but their potential in system requirement analysis remains unrealized. This study will look at how different large language models (LLMs) handle the subtleties of system design in order to give developers a better answer. One method focuses on clearly grouping system parts into categories, while the other looks at system parts that are more subtle and complicated.

1.2 Research Objective

This research aims to assess and compare two distinct approaches in large language models (LLMs) for analyzing similarities and differences in systems based on design and requirement specifications. The focus is on identifying which method provides clearer clustering and which one captures deeper, more subtle characteristics. This research seeks to identify the advantages and disadvantages of each model in grouping similar systems and recognizing the intricate, subtle characteristics that set these systems apart. The evaluation will focus on the models' ability to process complex systems comparison process, which has typically been a manual and time-intensive task. This study will examine how GPT and RoBERTa can identify patterns, classify systems, and offer valuable insights into the connections among different system technologies by applying these models to system descriptions and requirements. Similar to engineering and product development, we will evaluate the models to demonstrate their efficacy in managing intricate systems where slight alterations are critical. The primary objectives are to examine the efficacy of GPT and

RoBERTa in clustering systems with similar attributes, focusing on their requirements and configuration. Ultimately, the findings from this research will help streamline system design processes by leveraging AI tools, thereby reducing time and improving accuracy in complex technical domains.

- Examining how each model addresses nuanced variations in complex systems and the implications for requirements engineering.
- Conducting a comprehensive evaluation of the models' efficacy to identify the optimal one for automated system analysis.
- Providing a framework for the application of LLMs in requirements engineering and system design, emphasizing their capacity to enhance the precision and efficiency of tasks such as system comparison. The objective is to identify the model capable of analyzing intricate system designs and requirements with optimal flexibility and precision.

Ultimately, the goal is to identify the model that demonstrates superior flexibility and precision in analyzing complex system designs and requirements.

1.3 Research Question

The research questions that this study attempts to resolve are along the lines of the objectives that were listed above and are:

- How effective are GPT and RoBERTa at clustering systems based on their design and requirement similarities?

- In what ways do GPT and RoBERTa differ in their ability to capture subtle distinctions between complex system features?
- Can LLMs, such as GPT and RoBERTa, automate the process of system comparison in a way that improves accuracy and reduces time?
- Which model is better suited for analyzing systems with diverse and intricate characteristics, and what are the reasons behind its superior performance?

Chapter 2

Literature Review

This chapter gives an outline of the important books in system design and requirement engineering. In Section 2.1, we explore how system resemblance and requirement engineering are connected. There is a section 2.2 that talks about the use of big language models in scientific areas like engineering and healthcare. In Section 2.3, we glance at the embedding methods that LLMs use. In Section 2.4, we compare how well RoBERTa and GPT do on jobs like system design and requirement analysis. In Section 2.5, we look at how LLMs are used in system design and requirement analysis. In Section 2.6, we look at how to compare systems using grouping methods like t-SNE and UMAP. In Section 2.7, we talk about the problems that LLMs have when they have to compare systems. Finally, in Section 2.8, we look at the chances and future trends for LLMs in system analysis.

2.1 System Similarity and Requirement Engineering

In requirement engineering, improving performance and ensuring that different components work well together largely depends on understanding how similar various systems are. Recent advancements in large language models, Particularly one model has greatly enhanced our ability to manage large datasets. Its use of masked language modeling significantly enhances text classification and system comparison, leading to clearer distinctions between system components. (Liu et al. 2019). This technique enables a more structured and distinct comparison of system features, which is critical for analyzing system requirements with precision. Through the utilization of extensive datasets and the refinement of essential training methodologies, RoBERTa has demonstrated enhanced performance relative to predecessor models like as BERT, especially across several assessment benchmarks (Liu et al. 2019). RoBERTa is a potential instrument for intricate activities, such as the effective arrangement of system needs.

NLP approaches, more broadly, have also been successfully used to convert requirement specifications into object-oriented design models. This process improves object identification and system classification, which is particularly valuable in fields where accuracy is crucial, as it facilitates a more structured comparison of various technologies (Shinde, Bhojane, and Mahajan 2012) (Vaswani et al. 2017).

2.2 Large Language Models in Technical Domains

Since models like BERT, RoBERTa, and GPT came out, LLMs have grown a lot and are now very important in the area of NLP. At first, these models were meant to make things like text analysis, question-answering, and language translation better. Because of how these models were made, especially the transformer-based model shown by Vaswani et al. Vaswani et al. 2017, much bigger datasets can be used and natural language processing tasks are done better (Vaswani et al. 2017). RoBERTa builds on BERT and improves this design by using more data and better training methods, which leads to better results on several measures (Liu et al. 2019).

In addition to conventional NLP tasks, LLMs have been utilized in specialized fields like engineering, healthcare, and system design. In the field of engineering, large language models are being utilized more frequently to analyze technical documents, assisting in the categorization of system requirements and the evaluation of various systems. Models such as GPT have been utilized to create systems that automatically examine requirement specifications, providing innovative methods to automate tasks that were once manual and labor-intensive. This enables engineers to produce insights and make decisions with greater effectiveness (T. B. Brown et al. 2020).

Additionally, LLMs have demonstrated their usefulness in the field of healthcare. These tools are employed to analyze extensive medical literature, extract pertinent information, and assist in diagnostics by examining clinical data. This shows how LLMs can apply their skills to specific technical areas, highlighting their versatility and strength as tools beyond just natural language processing (Lee et al. 2020).

The emergence of LLMs in technical fields indicates a change in the way these sectors handle intricate, text-oriented information. As their capacity to comprehend and produce text resembling human communication advances, their usefulness in specific domains is expected to grow even more.

2.3 Embedding Techniques in LLMs

Embedding techniques are at the core of how large language models (LLMs) like GPT and RoBERTa process textual data, enabling them to capture intricate semantic relationships between words. Unlike static word embeddings like Word2Vec or GloVe, which represent words in isolation, contextual embeddings used in models such as RoBERTa and GPT are dynamic and adjust based on the surrounding text. The Transformer design which is used by (Vaswani et al. 2017) and made it much easier for models to handle long-range relationships in text, is used to make these contextual embeddings.

In RoBERTa, masked language modeling plays a vital role in generating these embeddings by masking certain words in a sentence and forcing the model to predict them based on context. This approach guarantees that the model incorporates all the many interpretations and links among words. More realistic models for activities including system design and requirement analysis follow from this (Liu et al. 2019). These embeddings help one model to grasp difficult technical papers and system requirements. This makes it highly effective for tasks such as clustering and system comparison, where clear distinctions are essential. In contrast, another model's embeddings allow for more intricate understanding of system features, making it useful for capturing the deeper connections between complex requirements.

In the same way, GPT pre-trains on very large datasets using unsupervised learning, creating embeddings that are very good at catching both grammatical and semantic information. This is very important for getting a handle on the complicated technical details in system designs. GPT embeddings can assist you in tasks like gaining understanding from system documentation and comparing how similar systems are depending on demand criteria (T. B. Brown et al. 2020).

Comparable in system design and requirement analysis, embeddings from these LLMs enable comparison and grouping of requirements. System analysis employment are thus more efficient. These techniques have changed the way text-based data is handled, allowing one to automatically and more precisely evaluate expert papers.

2.4 RoBERTa and GPT: Comparative Studies

Transformer-based models like GPT and RoBERTa have revolutionized natural language processing NLP with their ability to understand and generate text. These models, built upon the Transformer architecture introduced by (Vaswani et al. 2017), utilize self-attention mechanisms to analyze complex textual relationships. However, their distinct architectural choices result in significant differences in how they process and interpret information.

GPT adopts a unidirectional decoder architecture, excelling in sequential tasks like text generation and summarization. Its training objective, causal language modeling, focuses on predicting the next token in a sequence based solely on preceding tokens. This approach enables GPT to generate coherent, contextually relevant text but limits its ability to capture relationships between non-adjacent tokens in a bidirectional context.

In contrast, RoBERTa employs a bidirectional encoder architecture, allowing it to consider both preceding and following context when analyzing text. This bidirectionality, combined with masked language modeling as its training objective, enables RoBERTa to capture richer, more detailed relationships within text. Optimizations over its predecessor, BERT, such as dynamic masking and larger batch sizes, further enhance its performance in tasks requiring contextual understanding and precise classification.

Table 2.1: Comparison of RoBERTa and GPT

Feature	**GPT**	**RoBERTa**
Architecture	Transformer Decoder (unidirectional)	Transformer Encoder (bidirectional)
Directionality	Left-to-right	Considers both past and future tokens
Training Objective	Autoregressive language Model	Masked Language Modeling
Embeddings	Broad and generalized	Precise and nuanced
Strengths	Text generation, thematic grouping	Classification, fine-grained clustering
Limitations	Limited contextual understanding	High computational cost

The different ways of clustering and classifying show that one model is better at tasks that need precise classification and clear clustering, while the other is better at tasks that need more nuanced interpretation of system features and for getting deeper insights. RoBERTa performs better than GPT at tasks requiring exact classification and semantic clustering (Liu et al. 2019) thanks in great part to strong masked language modeling and plenty of pre-training on vast datasets. For technical jobs like requirement engineering, where proper grouping of system components is crucial, RoBERTa is therefore more suited.

Conversely, GPT models—especially GPT-3—are fantastic in producing and summarizing text and in handling tasks requiring natural language generation (T. B. Brown et al. 2020). Although GPT handles text generation well, it typically falls behind RoBERTa in classification tasks due to its weaker ability to grasp fine-grained semantic relationships. Studies demonstrate that GPT's ability to generate coherent descriptions of system requirements makes it effective for system summaries, while RoBERTa dominates in the analysis of system specifications and the identification of patterns for system clustering (Lee et al. 2020).

In conclusion, while both models have distinct advantages, RoBERTa is favored for tasks that require deep contextual understanding, such as clustering and classification, while GPT is more suited for text generation tasks.

2.5 System Design and Requirement Analysis with LLMs

LLMs are increasingly applied in system design and requirement analysis since they can manage sophisticated text data. Particularly when it comes to reading through lengthy system design documents and converting textual requirements into structured models, these models open a whole new path to automate jobs that used to be done by hand. Contextual embeddings from one model and generative abilities from another have helped improve the understanding of system specifications, which is important for making system design tasks faster and more accurate (Liu et al. 2019) (Rocha 2020) (Vaswani et al. 2017).

In system design, LLMs have been used to automatically find important system parts and judge how efficient the design is. For example, LLMs help requirement engineering by automatically turning textual specifications into formal design models. This lets engineers find possible gaps or inconsistencies in requirements early on in the design process. Certain models work especially well for requirement clustering tasks, enabling the grouping of complex system features based on functionality (Shinde, Bhojane, and Mahajan 2012).

Also, case studies have shown what LLMs do in the assessment of technology. GPT has been used to look through collections of system design documents and find patterns that help people make decisions and make it easier to compare systems (T. B. Brown et al. 2020). These models assist engineers in quickly generating insights from large-scale documents, reducing the time needed to assess complex system designs. Another study highlighted RoBERTa's superior performance in identifying and categorizing technical requirements, making it highly effective in ensuring system coherence and reducing design errors (Lee et al. 2020).

As LLMs continue to evolve, their ability to support system design and requirement analysis will further expand, automating tasks that would otherwise be labor-intensive and improving the accuracy of system comparisons across technical domains.

2.6 Clustering Techniques in System Comparison

Many people use clustering methods like t-SNE and UMAP to see large amounts of data and find patterns in tasks that compare systems. These methods are essential for identifying similarities between systems when applied to extensive and intricate data, such as technical specifications or system requirements. t-SNE and UMAP facilitate the visualization and comprehension of the interrelationships within a system by reducing high-dimensional data into two or three dimensions, thereby enhancing interpretability. t-SNE is a non-linear dimensionality reduction technique that works really well for seeing clusters in very high-dimensional data. It is often used to look into how systems are put into groups based on how they work or how they are built. When comparing systems, t-SNE has been used to evaluate how various LLMs find patterns in system requirements and group them into coherent clusters. One problem with t-SNE, though, is that it can take a long time to run on very large datasets (Maaten and Hinton 2008).

UMAP, on the other hand, is a newer method that can compute more quickly and often keeps more of the data's global structure than t-SNE. This makes UMAP very useful for tasks that need to look at system similarities in real time, like comparing big system designs or checking out technological frameworks (McInnes, Healy, and Melville 2018). By using UMAP on tasks that compare systems with LLMs, it has been shown to be useful in showing clearly how different systems group together based on their needs.

When it comes to system similarity analysis These clustering methods are very important for testing how well models like RoBERTa and GPT can understand the details in technical documents when using LLMs. Engineers can see how system requirements are organized with t-SNE and UMAP. This helps them to better grasp how LLMs treat intricate technical data. UMAP balances local and global data structures, hence research indicates it is better for designing clustering systems. This makes comparison of high-dimensional systems more favorable (McInnes, Healy, and Melville 2018).

Combining these clustering techniques with LLMs helps engineers and researchers to learn a lot about system comparison chores. This helps one to better organize and examine difficult technical specifications.

2.7 Challenges in System Comparison Using LLMs

LLMs know a lot about system design and requirement analysis, but they still have to deal with a lot of problems. Misclassification of technical terms and failure to accurately capture the relationships between system components is a big problem. This is often because they were trained in general language, which makes it hard for them to understand domain-specific language (Liu et al. 2019). Furthermore, some LLMs struggle with high-dimensional data in system specifications, which can make it hard to group systems that are similar (Radford et al. 2019).

Clustering methods like t-SNE and UMAP can help you see how systems are similar, but they might make confusing clusters when used on technical data. These models can get it wrong when there are small changes in the system requirements, which leads to bad classification results (Maaten and Hinton 2008) (McInnes, Healy, and Melville 2018). Also, LLMs get confused a lot because technical language isn't always clear, so they need more domain-specific tuning to get better at system comparison tasks.

2.8 Opportunities and Future Trends

LLMs hold significant potential for expanded applications in areas like comparing systems and figuring out what needs to be done as they get better. LLMs are becoming more important for automating complex system comparisons as these models get better at understanding contexts that are unique to a domain.(Liu et al. 2019) (Tăbușcă, Cojocariu, and Dobrescu 2023) suggest that LLMs are likely to be used in the future in specialized areas like engineering, healthcare, and aerospace to make system design and analysis easier.

These combined techniques will help models address challenges related to understanding domain-specific requirements and producing more accurate system comparisons. This mix can help you understand how systems work together better, which makes it easier to compare systems (McInnes, Healy, and Melville 2018). Also, making LLMs work better with domain-specific data will keep making them useful in technical fields, making sure they can pick up on the details of complicated systems (Tăbușcă, Cojocariu, and Dobrescu 2023).

Another area of opportunity is making LLMs easier to understand. These models are very useful, but it's not clear how they make decisions, which is a problem in important areas like engineering and system design. Models that are easier to understand and can explain how they classify and compare systems may be the focus of future research. For high-stakes events, this would increase their dependability (T. B. Brown et al. 2020).

LLMs improve with time in fast analyzing vast volumes of complex technical data. Future developments most likely will concentrate on improving their handling of challenging system requirements. This new development creates fresh approaches to investigate multi-modal models including not only text but also structured data such as blueprints or diagrams. This increases the value of LLMs even in system analysis (Tăbușcă, Cojocariu, and Dobrescu 2023).

Chapter 3

Environments and Background Theory

This section details the experimental setup, including the libraries, frameworks, and software environments utilized. It also provides background information on large language models (LLMs) and describes the various approaches employed in the experiments.

3.1 Environment and Software

3.1.1 Jupyter Notebook and Google Colab

For the practical components of this thesis, which involved API calls to GPT models and Python for visualizing and analyzing the results, Jupyter Notebooks[1] hosted on Google Colab[2] were utilized as the

[1] https://jupyter.org/

[2] https://colab.research.google.com/

primary environments for development, experimentation, and analysis.

Jupyter Notebook Jupyter Notebook is an open-source, web-based interactive environment that lets you run code, see instructions, and see data in a flexible way. Its design lets it run in modules or cells, which means that users can write and run code, record processes, and see the results all in one place. Things like pre-processing data, making API calls to GPT and RoBERTa, and showing results with tools like Matplotlib and t-SNE were split up into their own cells. This modularity not only made it easier to find and fix bugs, but it also made it faster and easier to test and change code at different points of the study. Jupyter's interface supported smooth transitions between experimentation and presentation, contributing significantly to the productivity of the research workflow[3].

Google Colab Google Colab extends the functionality of Jupyter Notebook to the cloud and provides free access to GPU and TPU resources, which are critical for handling the computational demands of large models like GPT and RoBERTa. Colab's cloud technology also makes it easy to scale up studies, which means that big datasets can be used to train and test models with little setup. A important part of this setup was also the merging of Google Drive, which made it easy to store, retrieve, and share big datasets and results. This feature was especially helpful for keeping track of system requirement data and making sure that all trial data could be accessed and shared in real time while the experiments were being done. Colab was an essential tool for running complex LLM-based tests quickly because it had cloud storage and powerful computer resources[4].

[3]https://link.springer.com/book/10.1007/978-1-4842-4470-8
[4]https://doi.org/10.1186/s40537-022-00565-9

3.2 Background Theory

3.2.1 Engineering Requirements and System Descriptions

In requirements engineering (RE), engineering requirements are very important for deciding how a system is designed, how it works, and what limits it has. They take the needs of stakeholders and turn them into solutions that can be tested. This makes sure that the system does what it's supposed to do. Most of the time, these rules come from official sources, like the NASA Systems Engineering Handbook or the International Council on Systems Engineering (INCOSE). These rules give clear explanations and steps for writing down and recording requirements. They use exact words like "shall," "should," or "will" to make sure everything is clear and can be tested (NASA 2016) (International Council on Systems Engineering (INCOSE) 2015). We have observed the potential for joint embedding to be used on system requirements (Bowen et al. 2024; C. L. Carroll et al. 2024; Chen, C. Carroll, and Morkos 2023; Chen, Wei, and Morkos 2023) and this research aims to contribute to this foundational work.

Design vs. Requirements:

System descriptions and requirements are very different. Needs are prescriptive and spell out what the system is supposed to do or achieve (Patel et al. 2024; Chen and Morkos 2023; Hein, Kames, et al. 2022; Hein, Kames, et al. 2021) , while system descriptions explain how the system is put together, what it does, and what features it has (Smith and Doe 2020). It is important to know the difference between these two types of systems because system descriptions often act as a link between official requirements and the finished product, including parts that weren't clearly stated in the original requirements. The variability in how system descriptions are interpreted can lead to different design approaches.

Functional vs. Non-Functional Requirements:

Functional requirements specify what a system must accomplish, such as particular behaviors, tasks, or capabilities (Shankar, Morkos, Yadav, et al. 2020; Hein, Voris, and Morkos 2018; Hein, Menon, and Morkos 2015; Shankar, Morkos, and Summers 2010). For example, these requirements may define how the system processes data or interacts with users.

Non-functional needs, on the other hand, spell out how the system does these things, including things like speed, dependability, security, and the ability to grow (Jones and M. Brown 2019). When comparing and grouping systems, both functional and non-functional needs are very important because they determine how the systems are built and how they work. System designs are different for each set of requirements and must be properly analyzed and grouped to make sure the system meets the needs of all stakeholders. When comparing and designing systems, engineering needs play a key part in how clustering and analysis are done. Knowing the differences between standards and system outlines helps you figure out how different systems work with their intended goals and in certain situations. Prior work has utilized transformers to determine relationships between functional and nonfunctional requirements (Mullis et al. 2024)

3.2.2 Transformer Architecture

The Transformer model, introduced by (Vaswani et al. 2017), revolutionized natural language processing and machine learning by replacing the reliance on recurrent neural networks (RNNs) and convolutional neural networks (CNNs) with a purely attention-based mechanism. This self-attention mechanism allows the model to capture relationships between different elements of an input sequence, regardless of their distance, making it ideal for tasks that require understanding long-range dependencies, such as system design and requirement analysis.

Self-Attention Mechanism The self-attention mechanism is the core component of the Transformer. In traditional sequence models, like RNNs, tokens are processed sequentially, making it difficult to capture long-range dependencies. In contrast, self-attention allows each token in the input to attend to every other token, enabling the model to weigh the importance of each token with respect to every other token in the sequence. In the context of system requirement analysis, this is particularly useful as it allows the model to capture complex relationships between different components of a system, which might be spread across a long technical document. For instance, in a large system, requirements in one section may be influenced by decisions made in another, distant section, and the Transformer can capture these dependencies efficiently. **Encoder-Decoder Structure:** The Transformer consists of two main components: the Encoder and the Decoder. Each of these components contains multiple layers of multi-head self-attention mechanisms and feed-forward networks.

Encoder: The encoder is responsible for processing the input sequence. Each layer of the encoder includes: Multi-head Self-Attention: This mechanism allows the model to focus on different parts of the input sequence in parallel, helping the model capture different aspects of relationships between the tokens. **Feed-Forward Networks:** After applying self-attention, the output is passed through a fully connected feed-forward network, allowing for non-linear transformations that further process the input. **Decoder:**Based on how the encoder's data was handled, the decoder makes predictions. In some ways, the framework is the same as the encoder, but it pays more attention to the encoder's output. This enables the decoder focus on the most important bits of the input while it makes the output order. When it comes to system design, the encoder can look at system specs to figure out how the different parts fit together, and the decoder can use the input to come up with insights, summaries, or labels.

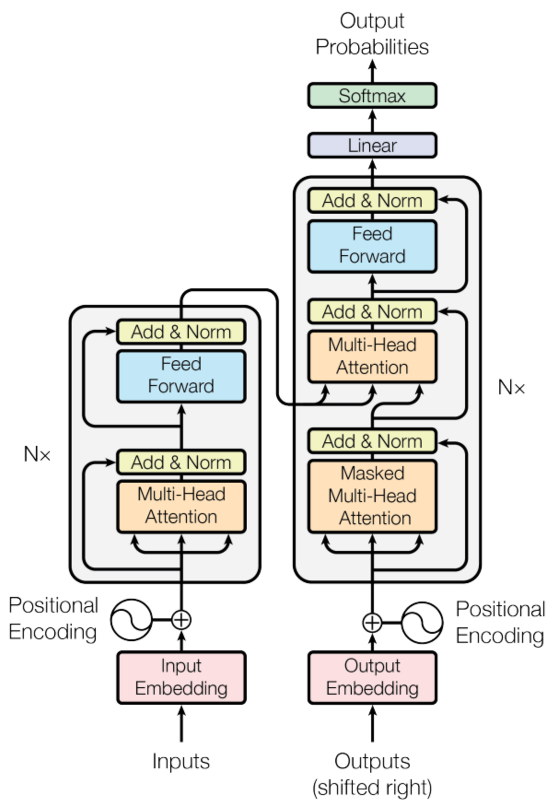


Figure 3.1: The Transformer - model architecture (Vaswani et al., 2017)

(Figure 3.1) will illustrate the Transformer's ability to capture dependencies in system design documents, enabling complex tasks like clustering and system comparison.

Advantages of Transformers

Parallel Processing: Transformers can process tokens simultaneously, rather than sequentially, making them significantly faster than RNNs, particularly for long sequences of data. Long-Range Dependency Handling: Transformers can capture relationships between distant elements in a sequence, making them ideal for analyzing lengthy system documentation. Scalability: Transformers can be scaled to handle very large datasets and complex models, such as GPT and RoBERTa, which are built on the Transformer architecture.

3.2.3 Clustering Techniques: UMAP and t-SNE

Clustering methods like t-SNE (t-distributed Stochastic Neighbor Embedding) and UMAP (Uniform Manifold Approximation and Projection) are needed to see high-dimensional data, especially embeddings made by LLMs. To find out how close two or more data points are to each other, these methods map the high-dimensional embedding space to a lower-dimensional space, which is usually 2D or 3D. In this case, the data points are system components or requirements.

t-SNE stands for "t-distributed Stochastic Neighbor Embedding."

(Maaten and Hinton 2008) came up with t-SNE, which is a common way to reduce the number of dimensions in data that you can see. As such it tries to preserve the local data structure by minimizing the divergence between a probability distribution reflecting pairwise similarity between original high-dimensional space and low dimension datas pace.

- **Local Clustering:** t-SNE does very well at clustering similar data points together, hence is good in noticing smaller clusters within the data.In this way, t-SNE can organize requirements-based parts into clusters that help with system design. This lets system engineers see how the different parts of the system work together.

- **Non-linear Dimensionality Reduction:** t-SNE finds non-linear connections between data points, which is very helpful when working with complicated data connections like those found in system requirement analysis.
- **Limitations:** t-SNE is good at showing local structures, but it can have trouble keeping global structures. This means that the lengths between groups might not be shown correctly. t-SNE also uses a lot of computing power, especially for big datasets, and can be affected by hyperparameter setting and other changes to the parameters.

UMAP stands for "Uniform Manifold Approximation and Projection."

A newer dimensionality reduction method called UMAP was created by (McInnes, Healy, and Melville 2018). It is used to show high-dimensional data in smaller dimensions, just like t-SNE. However, UMAP is faster and better at keeping the global layout than t-SNE in a number of ways.

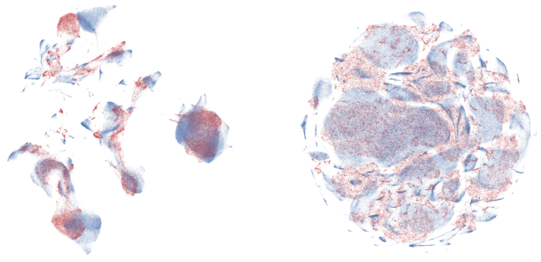
- **Local and Global Structure:** UMAP is better than t-SNE at keeping both local and global structure. This means that it not only groups data points that are similar together, but it also keeps the gaps between clusters better. This makes it very useful for jobs that need to know both small-scale relationships (within clusters) and large-scale relationships (between clusters), like comparing systems.
- **Faster Computation:** UMAP uses less computing power than t-SNE, so it can be used with big datasets and in situations where speed is important. In this study, large-scale system data will be handled, requiring fast and efficient clustering methods to maintain smooth workflows. This will assist in identifying similarities and differences in system designs.

- **Topology-Preserving:** One of UMAP's main advantages is that it is grounded in mathematical theory related to manifold learning and topology preservation. This means that UMAP is better at keeping the global shape of the data. This makes it better for big, complicated datasets where it's important to keep both local and global connections.

Comparison Between t-SNE and UMAP

- **Local vs. Global Structure:** UMAP preserves both local and global structures more effectively than t-SNE, which tends to prioritize local structures. When comparing systems, it's important to know both the small relationships within groups and the bigger relationships between them. UMAP is better at this.
- **Computation Time:** UMAP typically outperforms t-SNE in speed, particularly with extensive datasets, rendering it a more appropriate option for real-time visualizations or for handling huge system design datasets.
- **Interpretability:** t-SNE may distort inter-cluster distances, complicating the interpretation of global links, whereas UMAP's methodology better preserves the relationships between clusters.

(Figure 3.2) shows how the Procrustes distance compares UMAP and t-SNE embeddings when the dataset is sub-sampled. This distance shows how stable the embeddings are. We are going to compare a 10% subset of the dataset (shown in red) to the whole dataset (shown in blue). The Procrustes distance (the distance between the embeddings) goes down as the subsample size goes up. This shows that the grouping structures are stable. It shows that UMAP has more stable results than t-SNE for big datasets.



(a) UMAP (b) t-SNE

Figure 3.2: UMAP vs t-SNE - Procrustes-based alignment of a subsample (`https://arxiv.org/pdf/1802.03426.pdf`)

This is because UMAP gets a lower Procrustes distance. When comparing different system designs or needs, this stability is important because consistent embeddings help make sure that small changes in data don't cause results that are very different from what was expected. The figure also shows that t-SNE needed more iterations to reach convergence, which sped up the process but made it more stable.

Chapter 4

Methodology

4.1 Framework: Embedding-based system similarity analysis using GPT and RoBERTa

The main steps in analyzing system similarity with embedding models like GPT and RoBERTa are described in this framework. In the subsection that that follows, each stage will be covered in detail:

Data Collection:

The process begins with collection system descriptions and requirements data from various sources like user manual and technical documents.

The dataset is devided into three main categories:

Original System Description Dataset: Captures new and innovative an original system designs like iPos, CD Player.

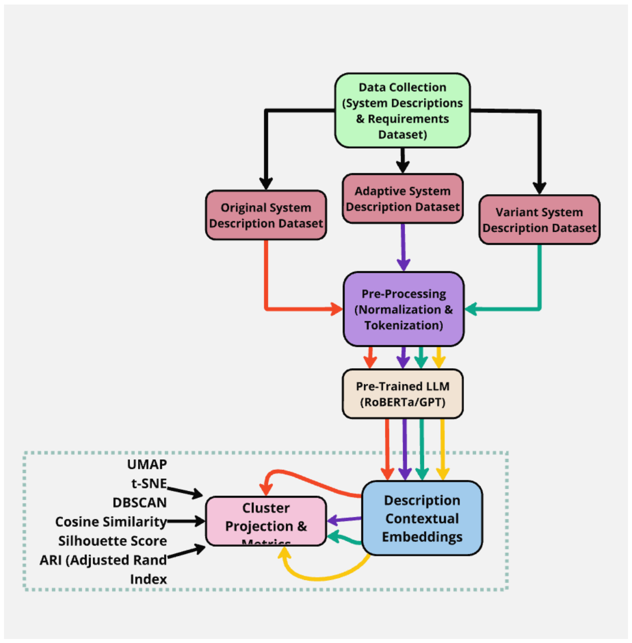

Data Collection (System Descriptions & Requirements Dataset)
Original System Description Dataset
Adaptive System Description Dataset
Variant System Description Dataset
Pre-Processing (Normalization & Tokenization)
Pre-Trained LLM (RoBERTa/GPT)
Description Contextual Embeddings
Cluster Projection & Metrics
UMAP
t-SNE
DBSCAN
Cosine Similarity
Silhouette Score
ARI (Adjusted Rand Index

Figure 4.1: Framework for Embedding-Based System Similarity Analysis Using GPT and RoBERTa

Adaptive System Description Dataset: Focuses on systems adapted for different functionalities like smart glasses, floppy disk.

Variant System Description Dataset: Includes systems with slight modifications or updates to existing designs like headphones, usb.

Pre-Processing (Normalization and Tokenization):

Data cleaning and preparation are conducted to ensure consistency, which converts text into lower case, removes irrelevant or unwanted symbols, and makes the text without any flaws.

Text is normalized (e.g., by removing irrelevant symbols) and tokenized text or split the data for embedding generation, which prepares data to generate the embeddings. This ensures compatibility with the pre-trained language models.

Embedding Generation Using Pre-Trained LLMs (RoBERTa/GPT):

Preprocessed data feeds into this phase as input to generate the embedded Contextual embeddings are generated using large language models, capturing semantic and structural information from the dataset.

Description of contextual Embeddings:

The embedded systems are analyzed to understand patterns and similarities between systems.

In this phase, embedding is generated from general knowledge and user-friendly text, which will improve accuracy for the specific task.

Also, this phase evaluates how well the embeddings represent the underlying data and their role in clustering results.

Cluster projection and metrics:

Dimensionality reduction techniques such as t-SNE(local structure) and UMAP(both local and global structure) are applied to visualize high-dimensional embeddings.

Clustering algorithms like DBSCAN are used to group systems with similar requirements or features.

Evaluation metrics like Cosine Similarity(for text similarity), Silhouette Score(for cluster quality), and Adjusted Rand Index (ARI) (for cluster quality accuracy) are employed to assess the quality of clustering.

4.2 Data Collection

In this study, system design and requirement data were gathered from a variety of publicly available sources, categorized to reflect the different degrees of novelty within the systems. This ensured that the collected data covered a broad spectrum of system designs, ranging from original designs to adaptive systems, which enabled a comprehensive analysis of both system descriptions and requirements.

4.2.1 System Descriptions

Based on how new they were, the system details that were gathered for this study were put into three main groups: original design, variant design, and adaptable design. Each group includes a variety of technologies and goods to make sure that the dataset for system analysis is broad and complete.

- **Original Design Data**

  This category includes systems that are completely new technology advances, where the design of the system brings new features or ideas. The original design data for this study came from a number of places, such as flight guides for NASA space missions like the Titan, Saturn, and Shuttle package delivery systems. Also, user guides for things like computers, iPods, CD players, tape players, and CD players were gathered. These systems are examples of goods that brought new technologies to their fields at the start, which had a big impact on those fields (Kapurch 2010).

  Digital cameras, smartphones, flat-screen TVs, and thermostats are some other examples of products in this group that changed their markets significantly because of their new designs.

- **Variant Design Data**

  Systems that have gone through small changes, like updates or modifications to an already existing product, fall into the variant design category. These systems have small changes to how they look or work, but they still follow the main design concepts of the original system. For instance, information was taken from USB devices, locks, door locks, and smart glasses, which show small differences in how safe or easy to use they are. Devices like trackers, headphones, and regular glasses are also in the collection for variant design. These have changed over time to add new features while keeping their basic design elements.

  These variations in design provide insights into how systems adapt over generations, capturing evolutionary product designs (Carrol 2024).

- **Adaptive Design Data**

  The adaptable design data includes items that have been changed to be used for completely different things. These systems show how ideas that are already in place can be changed or adapted to fit new requirements. Data from old cell phones that have been turned into smartphones and film cameras that have been turned into digital cameras are two examples. In the same way, manual typewriters have been turned into digital word processors, showing how design has changed from using traditional to digital technologies.

  CTTV (closed-circuit television) and AR glasses have also been studied as adaptable systems. CTTV has become a modern monitoring device, and AR glasses have become an augmented reality device. Reusing old technology in new ways to meet new market needs or functional needs is an innovative idea that these adaptable systems show (Smith and Doe 2020).

4.2.2 Dataset Overview:

Data were collected and categorized based on system novelty, focusing on requirements and system descriptions across diverse technological designs. Table 4.1 presents a summary of the dataset statistics, organized by Original, Variant, and Adaptive Design categories.

Explanation of Dataset Statistics

Table 4.1 provides a statistical summary of the requirements dataset across the three design categories. Each column serves a distinct purpose in understanding system complexity and design evolution:

- **Number of Requirements:** This column indicates the quantity of documented requirements per system, providing a measure of the system's complexity.
- **Avg. Words Per Requirement:** This shows the average word count per requirement, offering insight into the detail or specificity used to specify each system's requirements. A higher word count may indicate more detailed or complex requirements.
- **Vocabulary Size:** This column captures the unique vocabulary used in each system's requirements, reflecting the linguistic variety and the presence of technical terms. A larger vocabulary size may indicate a more complex or specialized system.

Together, these statistics highlight differences in requirements across systems and design types.

Table 4.1: Requirements Dataset Statistics

System(s)	**Number of Requirements**	**Avg. Words Per Requirement**	**Vocabulary Size**
	Original Design		
Film Camera	13	12.77	116
Digital Camera	15	12.33	129
iPod	48	15.40	262
CD Player	33	21.06	354
Cassette	25	17.96	223
Total	**134**	**15.90**	**1,084**
	Variant Design		
Smart Glasses	22	22.05	251
Headphones	21	25.43	273
Traditional Glasses	12	18.42	131
AR Glasses	15	20.93	180
USB	10	14.10	83
Total	**80**	**20.19**	**918**
	Adaptive Design		
Typewriter	13	14.85	105

System(s)	Number of Requirements	Avg. Words Per Requirement	Vocabulary Size
Computer	16	12.06	116
Mechanical Watch	12	14.92	98
Smart Watch	12	13.92	107
Manual Door Lock	10	10.10	81
Smart Door Lock	13	11.54	113
Manual Thermostat	14	11.07	101
Smart Thermostat	17	11.88	124
Floppy Disk	12	12.75	83
Total	**119**	**12.67**	**928**

4.2.3 System Requirements

In addition to system descriptions, system requirements were gathered from both documentation and internal sources. The dataset includes technical specifications for a variety of systems, including telescope components and complete telescopic systems, sourced from technical manuals and system engineering documents.The collection was made even better by adding in-house data about the needs of industrial systems like a space shuttle system and a smart door lock that made sure it included a wide range of space and consumer technology systems.

This multi-layered approach of gathering data, which includes different system designs, makes sure that the comparison of similar systems is thorough and can be used in many different technology areas. To keep the focus on system and requirement descriptions that are directly related to design and performance, unnecessary data like project descriptions or data that wasn't related to how the system works was left out. This study successfully covers both functional and non-functional requirements across a wide range of businesses by putting together a wide range of system descriptions and requirements. Consumer gadgets to aircraft parts were studied, and the results can be used in a lot of different areas. This means that the insights can be used in a lot of different technological situations (Carrol 2024) (Smith and Doe 2020).

4.3 Pre-processing and Model Embeddings

The pre-processing part of this study was very important for getting system descriptions and needs ready for embedding generation. The organized method included normalization, tokenization, and embedding creation, which made it possible to analyze system data in great detail. For this, two models were used, and each one added something different to the anchoring process.

4.3.1 Data Structuring and Grouping

There were three groups of system definitions and requirements:

- Requirements across systems
- Descriptions categorized by novelty
- Combined requirements-descriptions sets

These groups were broken up into single lines to allow for more detailed analysis, which showed how the system worked and what its design needs were (Carrol 2024).

4.3.2 Tokenization

As the first step in embedding creation, tokenization, the text was broken up into smaller pieces, such as words, sub-words, or characters. Each of these pieces was then given a unique token ID. This step was very important for understanding the system descriptions:

- RoBERTa's tokenization preserved complex relationships by capturing the social meaning of words, even those that can be interpreted in more than one way (Devlin et al. 2018).
- The other model used a similar method to make sure that the inputs kept coming in the right order, which is important for figuring out the structure and flow of system needs.

4.3.3 Normalization

Normalization was used to clean up the text by getting rid of symbols that weren't needed, making sure that all the words were in the same case, and getting rid of stopwords when they were needed. This made sure that both models only got knowledge that was useful, focused on the most important parts of system design and requirements.

4.3.4 Embedding Generation

After tokenization and normalization, the models made embeddings, which turned the words into vectorized representations. These embeddings caught both the meanings and the structures of the words:

- In the case of RoBERTa, the text that had been handled was turned into tensors and sent through a Transformer design. This created embeddings that were rich in context. The model that had already been trained didn't get any extra tweaks. other model used the same embedding method and did a good job of catching sequential relationships and contextual flow within system descriptions. As with the first model, there was no fine-tuning done to keep the knowledge that had already been learned (Devlin et al. 2018).

Types of Embedding and Positional information: There were three main types of embeddings that were made:

- character embeddings showed what each character meant in its setting.
- Positional embeddings kept the order of the words in a sentence.
- Segment embeddings showed how different parts of a sentence or sentences are related to each other.

Input sequences for both models had a classification symbol that summed up the whole sequence. This made a 768-dimensional vector for each sentence. After that, these vectors were saved for jobs like comparison and grouping (Carrol 2024).

4.4 Data Analysis Methods

A mix of clustering and dimensionality reduction methods were used to look for parallels between systems and make high-dimensional data easier to understand. These methods made it easy to compare system needs and descriptions in detail, which revealed trends and connections that weren't obvious from the raw data alone. We used t-SNE (t-distributed Stochastic Neighbor Embedding) and UMAP (Uniform Manifold Approximation and Projection) as the main tools to reduce the number of dimensions and show the embedding space.

t-SNE: Non-linear Dimensionality Reduction A lot of people use t-SNE to see high-dimensional data by shrinking it to two or three dimensions. t-SNE is better than standard linear methods like PCA at keeping local structure, which means that it keeps the relative lengths between points that are close to each other in a high-dimensional space.

In this study, t-SNE on the contextual embeddings from system descriptions and requirements in this study to see how systems group together based on how similar they are. t-SNE helped us understand local clusters by finding the pairs of data points that are most similar and reducing the amount of difference between the original high-dimensional space and the reduced two- or three-dimensional space.

- **Advantages:** t-SNE is great at finding local groups, which is especially helpful for finding small differences in how systems work and what they need to do (Maaten and Hinton 2008).

- **Limitations:** t-SNE keeps local connections but not so good at keeping global structure. Also, it can be hard to use on big datasets because it takes a lot of time and resources (McInnes, Healy, and Melville 2018).

In this study, t-SNE showed how the systems were grouped together based on their design and needs, pointing out the ones that had similar features.

UMAP: Efficient and Topology-Preserving UMAP, a more recent dimensionality reduction technique, offers a balance between local and global structure preservation. It works by putting together a high-dimensional graph of data points and finding the best way to lay them out in a lower-dimensional area. Because of this balance, it is very helpful for understanding both small groups and the connections between systems as a whole.

- **Advantages:** UMAP can handle bigger datasets better than t-SNE because it can do calculations faster. It also tends to keep both local and global topology, which makes sure that relationships between faraway groups in the embedding space are shown more accurately (McInnes, Healy, and Melville 2018).
- **Applications:** In this study, UMAP added to t-SNE by giving a bigger picture of how systems work together, which helped people understand the information better as a whole. This was especially helpful when figuring out how systems with small differences fit into a bigger picture.

Cluster Analysis and Visualization t-SNE and UMAP were both very important in finding groups of systems that had similar needs or design traits. After reducing the number of dimensions, these groups were shown using Matplotlib, a Python tool for making 2D scatter plots. Each plot showed a system as a point, and groups showed systems that were alike based on how they were embedded in their surroundings. Different types of systems were distinguished by color-coding, which made things easier to understand visually.

By using t-SNE, UMAP, and Matplotlib together, it was possible to find trends in system similarities that were both local and global. These methods gave important information about how systems were similar and different in many ways, including their needs and designs.

4.5 Evaluation Metrics

Various significant metrics, including the silhouette score and the adjusted rand index (ARI), were employed to evaluate the effectiveness of the clustering results and system comparisons. These measures assisted in determining the quality and consistency of the groups generated by the models and the DBSCAN algorithm.

4.5.1 Silhouette Score

The shape score evaluates the degree to which each data point aligns with its designated cluster in comparison to its alignment with alternative clusters. The score ranges from -1 to 1, with higher values indicating more distinct group definitions. A score near 1 indicates that data points are more distant from points in other clusters and nearer to points within their own cluster (Rousseeuw 1987).

- **Implementation:** The sklearn.metrics.silhouette_score function was employed to calculate the silhouette score for this study. The embeddings were utilized to evaluate the effectiveness of the models in categorizing system components or requirements.

- **Purpose:** The outline score showed that the groups were cohesive, making sure that systems with similar needs were put together correctly and systems with different needs were clearly separated (McInnes, Healy, and Melville 2018).

4.5.2 Adjusted Rand Index (ARI)

The adjusted Rand index looks at how similar two sets of data are by taking into account how the points are randomly placed. A score of 0 indicates random clustering, while a value closer to 1 suggests strong alignment between clusterings (Hubert and Arabie 1985).

- **Implementation:** The function sklearn.metrics.adjusted_rand_score was utilized to calculate the ARI. The primary objective was to compare the outcomes of various grouping experiments that were conducted multiple times.
- **Purpose:** The objective was to determine the level of consistency among the groups, particularly in relation to the outcomes of different DBSCAN clustering executions on the generated embeddings (Maaten and Hinton 2008).

4.5.3 DBSCAN Clustering

DBSCAN clustering was employed to identify groups of systems that exhibit similar requirements, despite not being a metric on its own. DBSCAN is a clustering method based on density that groups together dense clusters while identifying outliers as noise.

- **Purpose:** DBSCAN served as the foundation for categorizing system requirements. The quality of these groups was assessed through the silhouette score and Adjusted Rand Index (ARI), facilitating precise comparisons and classifications of the systems (Carrol 2024).

4.5.4 Visual Evaluation via Matplotlib

In addition to the numerical measures, scatter plots created with Matplotlib were employed to visually examine the clustering results. This qualitative study looked at how well the system's goals and design features were grouped by how well they fit with the system.

- **Implementation:** To see the DBSCAN groups for the embeddings, scatter plots were made. Making sure that systems with similar needs or traits were grouped properly through this visual check (Smith and Doe 2020).

4.6 System Similarity Assessment

The analysis examined both functional and non-functional needs through the application of embedding-based similarity measures and clustering methods to assess the similarities and differences among the systems. These methods provided us with a comprehensive understanding of the similarities and differences among various systems.

4.6.1 Assessing Similarity Derived from embeddings

The system descriptions were utilized to create embeddings that captured semantic relationships. This allowed for the comparison of systems according to both functional and non-functional requirements.

- **Contextual embeddings:** The requirements and specifics of the system were utilized to create these embeddings, which captured intricate contextual patterns that facilitated the identification of connections between systems. The models successfully captured both syntactic and semantic details by converting the descriptions into vectorized representations.

- **Cosine Similarity:** Cosine Similarity was utilized to assess the similarity between two system embeddings. (Carrol 2024) suggests that systems exhibiting higher cosine similarity values are considered more similar regarding their functional and non-functional characteristics.

4.6.2 Clustering Based Similarity

Based on their embedding models, systems were put into groups using clustering methods. This made it possible to find traits that all the systems had in common.

- **Cluster grouping:** Tools like DBSCAN were used to find groups of systems that had things in common. These groups facilitated the automatic categorization of systems with similar characteristics according to their functional and non-functional requirements (Smith Doe, 2020).

- **Functional and Non-Functional Requirements:** Systems can be categorized based on their functional requirements, which describe what the system accomplishes, and their non-functional requirements, which detail how those tasks are performed. This multi-level method enabled a more comprehensive comparison of systems.

4.6.3 A comparison of Visual and Quantitative

By employing both Visual and Quantitative methods, it became feasible to compare systems and assess their proximity to one another.

- **Visual representation:** Techniques such as t-SNE and UMAP allow visualization of the embedding space in two or three dimensions, resulting in systems with comparable functional or non-functional characteristics appearing closer together. Systems that exhibited greater dissimilarity appeared to be more distant, illustrating the functioning of global and local connections (McInnes, Healy, and Melville 2018).

- **Quantitative Comparative:** Cosine similarity scores and cluster cohesion were used to find the functional and non-functional parallels between systems in a measured way. This made sure that the visual groups matched the systems' real technical and performance features (Maaten and Hinton 2008).

4.6.4 Application in Requirement Engineering

This approach allowed engineers to determine, depending on demands, which systems were similar and which were distinct by combining visual and numerical approaches. These approaches made it simpler to compare systems, improve design decisions, and provide more knowledge about how comparable systems operate by combining like systems.

Chapter 5

Results

5.1 Overview of Experiment Outcomes

This analysis aimed to assess the effectiveness of the two NLP models, GPT and RoBERTa, in clustering technologies based on system criteria and their descriptions. Some categories of products involved in the dataset were the electronics consumer goods, including television, smartphones, cameras, and adaptative devices like augmented reality glasses and smartwatches. The goals of the experiments were to compare how well the three models captured semantic associations between technology items and discriminated between similar objects and technologies in the four categories.

In both cases, the models created semantic embeddings of the technologies, which were then analyzed using the DBSCAN algorithm. DBSCAN was chosen for its ability to cluster the data based on density and to locate outliers or noise points or, in other words, technologies that cannot be categorized under any of the groups. To investigate the relationship between the clusters and perform a structure analysis of these clusters, the data was visualized in two dimensions using UMAP and t-SNE methods. Quantifies the density of connections within a cluster, with higher values implying that technologies belonging to the same cluster are more related. Measures the extent to which the anticipated clusters match the actual categories, which indicates the effectiveness of the models in categorizing the technologies.

The results showed that RoBERTa always produced more compact clusters, translating into higher silhouette scores. It was most applicable for distinguishing between categories such as adaptive technologies (AR glasses in contrast to smart glasses) or variants. However, the larger clusters in GPT were apparent issues distinguishing between technologies with related uses. This was evident in the fact that there was a higher merging of categories that fell under consumer electronics, for instance, cameras and smartphones, where the functions could have been more distinct. Regarding noise management, fewer outliers were detected by RoBERTa (Cluster -1), which implied that RoBERTa could classify technologies under suitable clusters more effectively. GPT produced more noise points, especially in adaptivity, suggesting it had difficulty classifying certain items.

The next sections will analyze the cluster outcomes for both models using the graphs. This information will further enhance the understanding of each model's advantages and limitations in creating clusters of technologies based on system similarities.

5.2 Comparison of Clustering Results

5.2.1 General Grouping of Technologies

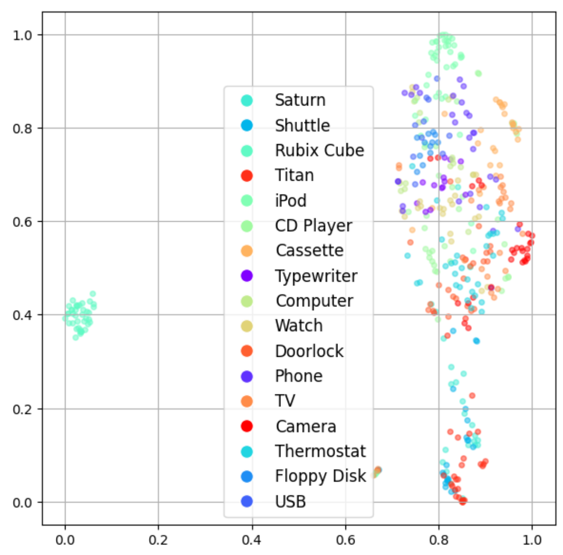
In this section, the focus is on the comparative performance of GPT and RoBERTa in categorizing technologies based on the underlying system characteristics. Each model's ability to cluster similar technologies (such as first releases, derivatives, and applications) was evaluated using semantic embeddings. Visualization techniques like UMAP (Uniform Manifold Approximation and Projection) and t-SNE (t-distributed Stochastic Neighbor Embedding) were employed to assess the cohesiveness and separability of these clusters.

In Figure 5.1(a), representing GPT's clustering, the plot is characterized by broader and less clearly defined clusters, with more overlap among products. For instance, technologies such as cameras and smartphones, which could belong to the same category in some aspects, were placed in the same cluster by GPT, suggesting that the model prioritizes semantic similarities over distinct functionalities. This overlap indicates that GPT's clustering approach leaned more on general thematic likeness, which limits its ability to create tightly defined clusters.

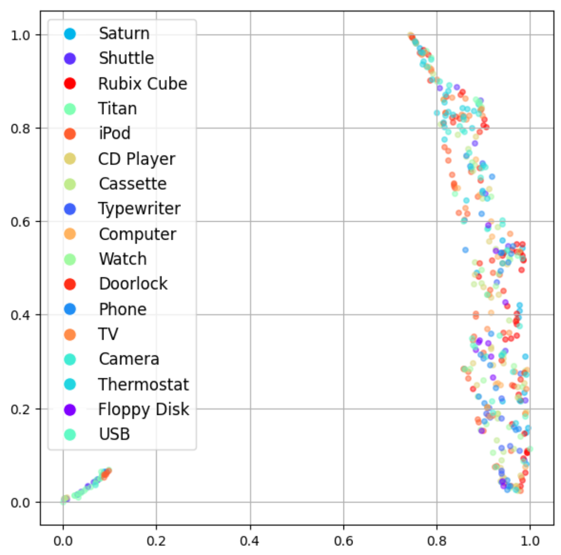
In contrast, RoBERTa's UMAP projection in Figure 5.1(b) displays distinct and compact clusters, illustrating its ability to identify and separate nuanced features. For example, AR and smart glasses—despite their functional overlaps—were placed in different clusters, reflecting RoBERTa's capacity to recognize minor differences in system requirements. This suggests that RoBERTa excels in fine-grained categorization, effectively distinguishing between products with similar descriptions for different uses.

These trends are corroborated by the t-SNE projections provided in Figures 5.9(a) and 5.9(b). The RoBERTa t-SNE visualization plot in Figure 5.9(a) exhibits well-separated and compact regions without much overlap, attesting to the texts' high agreement with product description categories. Smartwatches and thermostats were identified as belonging to different categories, which confirms that RoBERTa is better at sorting products based on the system's requirements.

Conversely, the t-SNE plot of GPT indicates that several clusters overlap, particularly between adaptive technologies and consumer electronics, as identified in Figure 5.9(b). For instance, AR and smart glasses were categorized together, explaining why GPT struggled to differentiate between similar products. This



(a) UMAP projection for RoBERTa



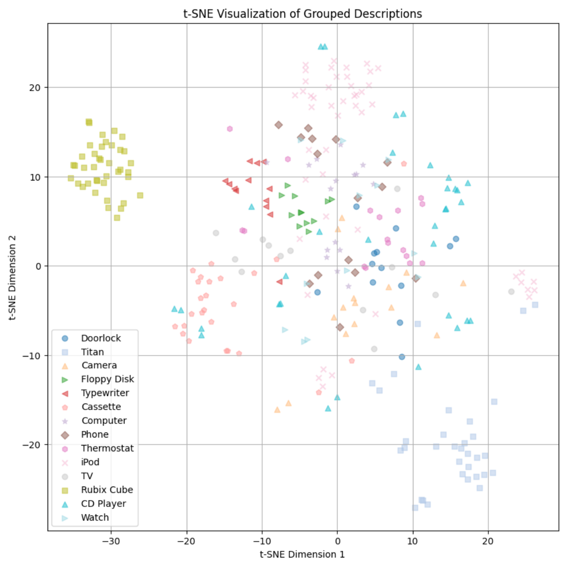
(b) UMAP projection for GPT

Figure 5.1: UMAP projections

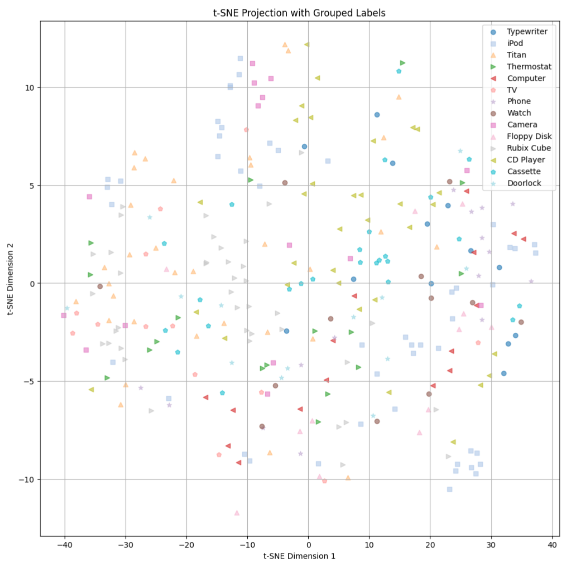
would imply that GPT focused on semantic relations more and may have placed less emphasis on subtle functional distinctions.

Analysis of noise handling and cluster size distribution: One identifies noise points during the cluster analysis and categorization of the data. Cluster -1 refers to the collection of Noise points, which means the models could not categorize these technologies appropriately. Roberta recognized fewer noise points, which indicates that most technologies were classified accurately into relevant clusters. Also, GPT created a higher number of noise points, and this was particularly evident in the context of adaptive technologies where functional interfaces between products were not demarcated.

The cluster size distributions (Figures 5.3(a) and 5.3(b)) also clearly show these differences. The distribution of technologies in Roberta is presented in Figure 5, indicating that most technologies are placed in meaningful clusters with fewer noisy or marginal technologies. Conversely, the distribution of GPT



(a) t-SNE projection for RoBERTa



(b) t-SNE projection for GPT

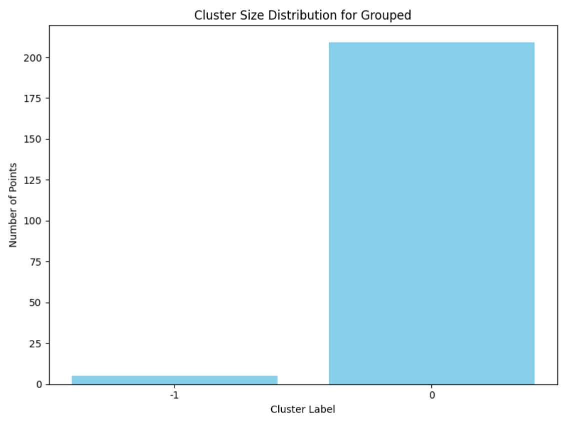
Figure 5.2: t-SNE projections

scores (Figure 5.3(b)) demonstrates a relative homogeneity of the scores across clusters and more significant outlying values. This implies that even though GPT helped to group technologies into broad blocks, it could have been more helpful in categorizing the products into closer categories.

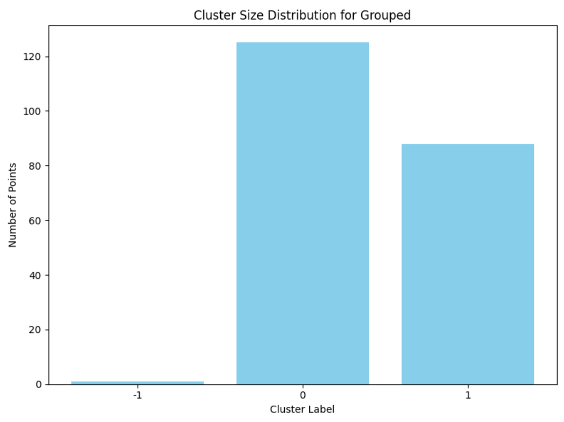
Summary of General Grouping Performance

In conclusion, Roberta outperformed the other methods in general clustering, making highly compact and well-separated groups that correspond to the product categories. Tighter clusters and less noise on the graph signify the programs' efficient handling of adaptive, original, and variant designs. The use of GPT in comparing the technologies was found effective in identifying technologies that were quite similar. However, it presented some difficulties in differentiating technologies that shared standard functions.

This reduced the ability to have clear and distinguishable clusters and produced more noise points and overlapping clusters. It points to the fact that the clustering solution proposed by Roberta helps identify



(a) Clustering size chart for RoBERTa



(b) Clustering size chart for GPT

Figure 5.3: Clustering Size

finer distinctions between the products compared to GPT for tasks in which a clear or coarse division is more than adequate. These insights provide the basis for the subsequent exploration of how each model operated in distinct technology groups and their capability to navigate nuances.

5.2.2 Grouping by Technology Type

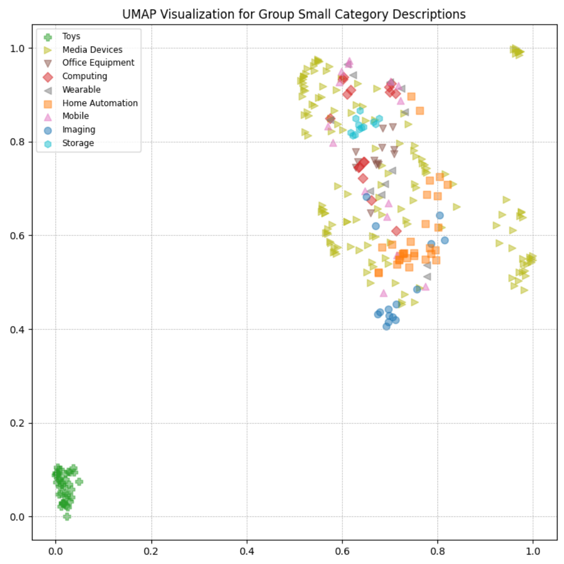
This section examines how GPT and RoBERTa models categorized technology based on its classification as mobile devices and toys. The goal is to compare and contrast the models' approaches to classifying products with similar descriptions but different uses.

Toys and Media Devices

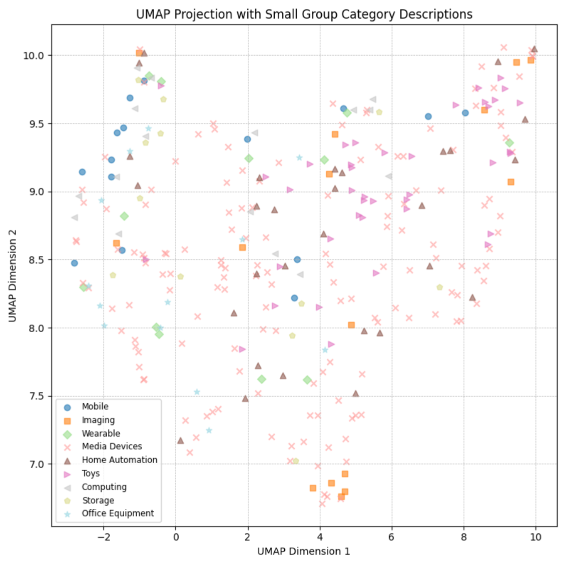
In Figure 5.4(a), the UMAP projection for GPT reveals overlapping clusters containing both media devices and toys. For example, products typically classified as media devices, such as TVs and speakers, are grouped near toys, indicating that GPT struggled to differentiate between these categories. This overlap suggests that GPT relied heavily on semantic similarities without sufficient distinction based on functionality.

Consequently, technologies with vastly different purposes, like entertainment devices and toys, were clustered together, revealing GPT's limitations in recognizing functional differences.

In contrast, Figure 5.4(b) shows that RoBERTa produced well-defined clusters, with a clear separation between media devices and toys. For example, media devices like TVs are distinctly clustered away from toys, illustrating RoBERTa's ability to categorize based on functional characteristics. This separation indicates RoBERTa's capability to recognize and group items with different uses accurately, making it more effective for tasks requiring detailed classification. The distinct placement of media devices and toys highlights RoBERTa's precision in categorization, especially in differentiating items that serve unique functions.



(a) UMAP projection for RoBERTa



(b) UMAP projection for GPT

Figure 5.4: UMAP projection for Group system

5.3 Analysis of Specific Technology Groups

5.3.1 Separate Grouping of Similar Technologies

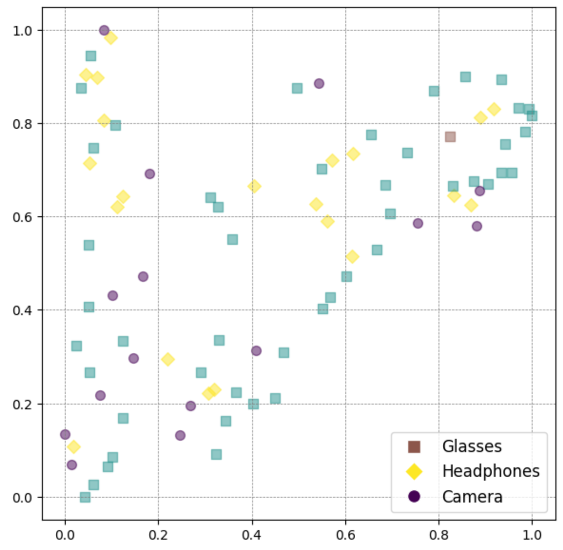
One of the main issues arising during the models' assessment was the models' capacity to cluster semantically similar but functionally distinct technologies. This was mainly examined with items such as AR and smart glasses, headphones and wearable devices, and manual and smart thermostats.

In GPT's projection, shown in Figure 5.5(a), AR/smart glasses and headphones are placed within the same cluster, indicating that GPT heavily relies on textual matching rather than functional categorization during clustering.

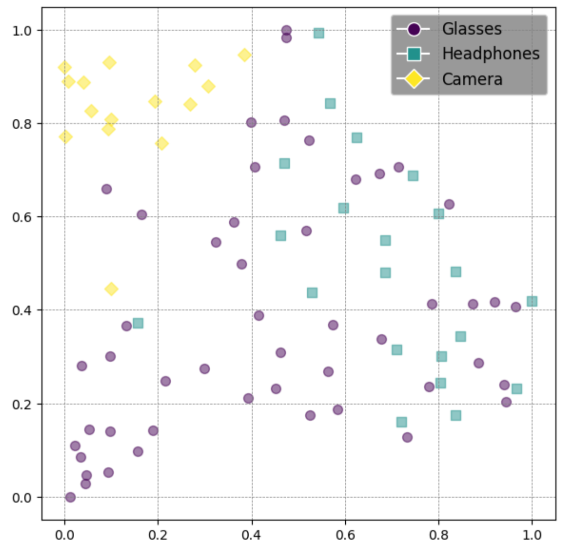
This clustering behavior demonstrates that GPT focuses primarily on the semantic overlap between technologies. As a result, products with slight functional differences, such as AR glasses (designed for augmented reality with audio and video) and camera (focused on video features), are mingled into the same cluster. This skewing highlights the limitations of GPT in distinguishing between technologies with marginal but important differences.

However, in the RoBERTa-generated projection, shown in Figure 5.5(b), these technologies are grouped more accurately. RoBERTa assigns glasses, camera and headphones to separate clusters, indicating that it considers both functional attributes and system-level requirements in its clustering.

This divide shows that RoBERTa is more effective at capturing minor functional differences, making it more suitable for categorizing technologies with overlapping descriptions but distinct purposes. As a result, RoBERTa provides a more reliable categorization of products into detailed and functionally meaningful clusters.



(a) UMAP projection for GPT



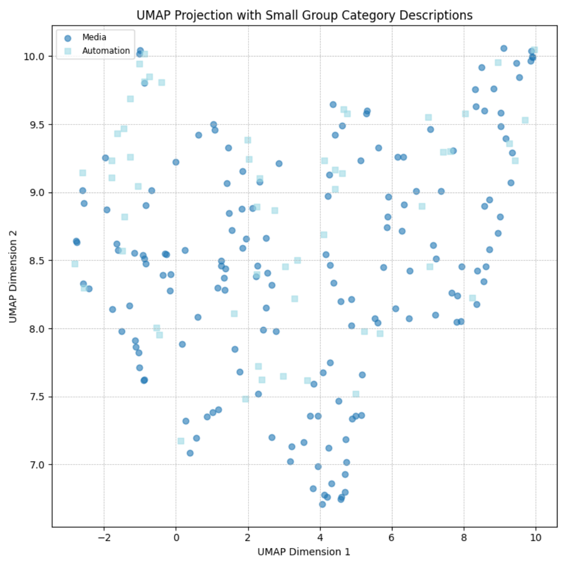
(b) UMAP projection for RoBERTa

Figure 5.5: UMAP projection for Media and Automation Group system

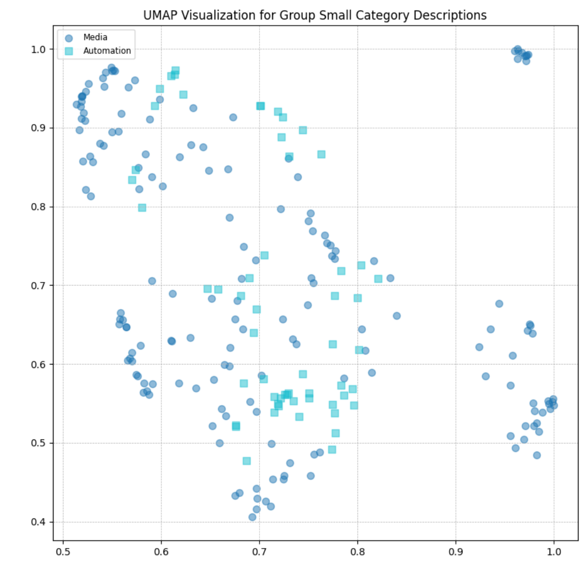
Furthermore, Roberta enhanced the differentiation in smart or media devices and automation, as illustrated in Figure 5.6(b). However, the grouping of GPT also revealed the overlapping of automation and mobile devices in Figure 5.6(a), which is even more suggestive of GPT's poorer performance in the fine differentiation of clusters.

5.3.2 Case Study: Adaptive Technologies – AR Glasses and Smart Glasses

A fascinating example is separating AR glasses and smart glasses—two adaptive interfaces that may be described similarly but differ significantly in their functions. AR glasses display a projected image over a real-world scenario, while smart glasses are mainly used for notifications and limited computing capabilities.



(a) UMAP projection for GPT



(b) UMAP projection for RoBERTa

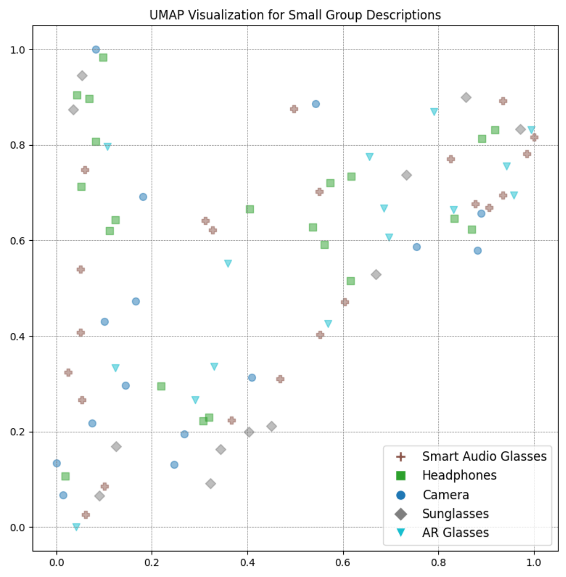
Figure 5.6: UMAP projection for Media and Automation Group system

In GPT's UMAP projection two types of glasses are clustered together, meaning that the model focused on the textual resemblance between product descriptions rather than differences in use cases. This led to higher noise and mismatch between the categories and ground truth categories.

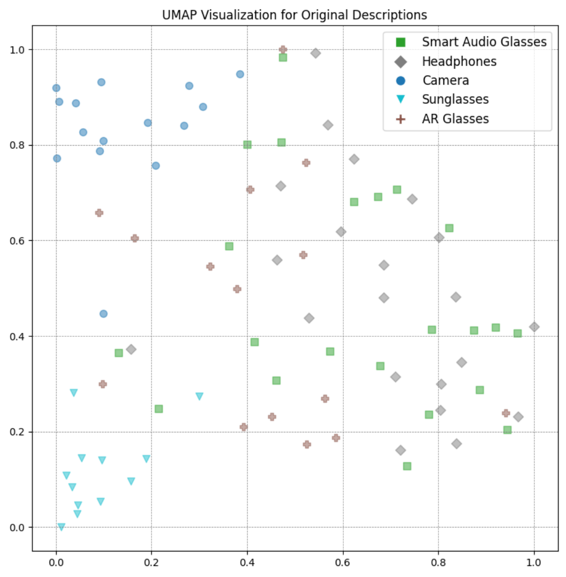
RoBERTa, however, handled this distinction much better. The model categorized AR and smart glasses into different groups and thus proved its proficiency in categorizing functional differences. This performance demonstrates how Roberta addresses concerns associated with complex product categories.

Summary of Findings

Regarding plotting technologies with non-interchangeable functions and significant degrees of retrieved features, such as RoBERTa, they are more effective than GPT. The fact that GPT forms more compact clusters leads to intersected categories, such as wearable and mobile devices or AR and smart glasses. On



(a) UMAP projection for GPT



(b) UMAP projection for RoBERTa

Figure 5.7: UMAP projection for Group system

the other hand, by classifying products depending on their system requirement, RoBERTa can form tighter and more accurate groups.

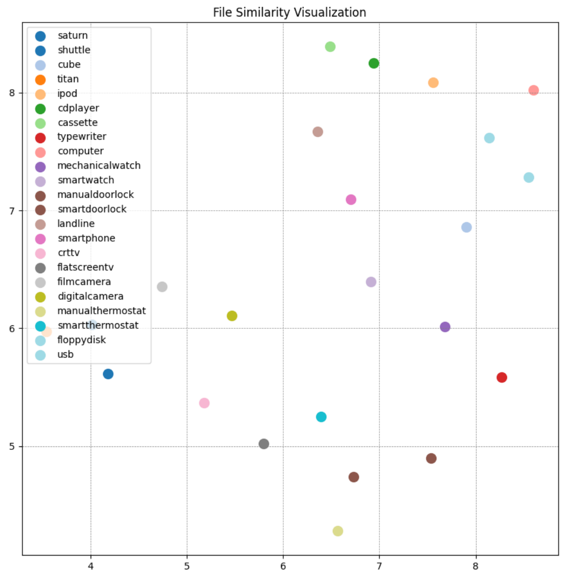
This evidence implies that RoBERTa is well-mannered for high-level product categorization use cases such as inventory control and recommendation engines. In terms of its use, GPT is more advantageous for simple clustering analysis investigations, where the generality of clustering is more important than the specificity of identities.

5.4 Graphical Analysis

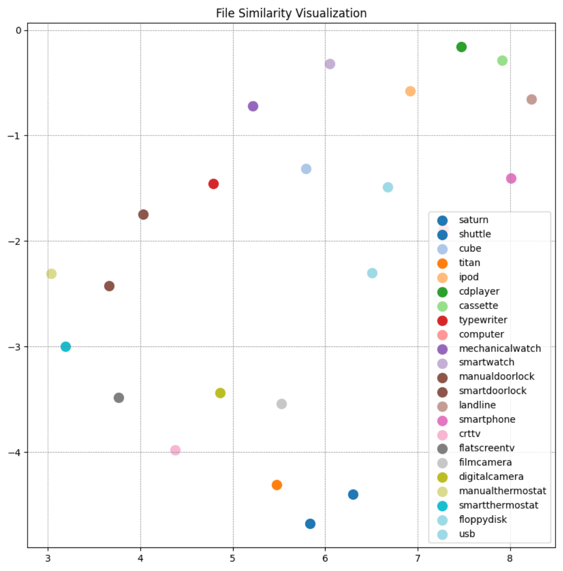
5.4.1 Visual Result Analysis with UMAP and t-SNE Projections

Visual Result Analysis with UMAP and t-SNE Projections This part talks about what we learned from UMAP and t-SNE visualizations and how GPT and RoBERTa put different technologies into groups based on their functions and how they work together.

UMAP Projections:



(a) UMAP projection for GPT



(b) UMAP projection for RoBERTa

Figure 5.8: UMAP projection visualization for the entire system's description

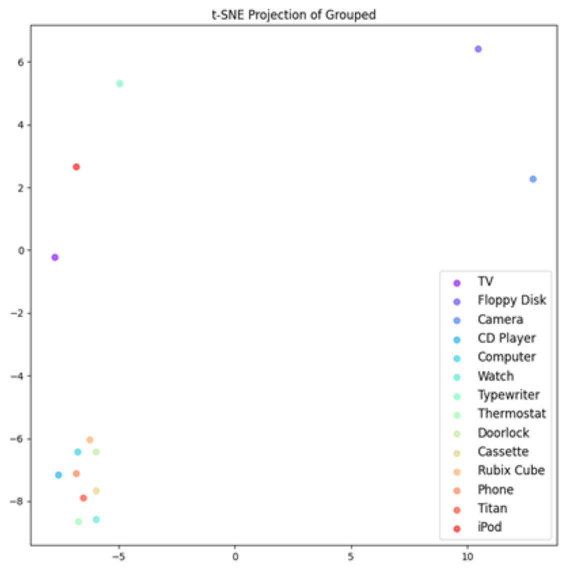
In Figure 5.8(b), According to RoBERTa's UMAP projection, there are clear and well-separated clusters for many different technologies, from common consumer electronics like TVs, cameras, and smartphones to new and flexible technologies like augmented reality devices and digital smartwatches. This separation emphasizes RoBERTa's ability to categorize technologies based on nuanced functional distinctions, capturing both conventional and adaptive items within clearly defined boundaries. The clusters for different groups, such as media devices and mobile devices, remain distinct, highlighting RoBERTa's capability to identify and group technologies according to specific uses or design criteria.

On the other hand, Figure 5.8(a) shows the plot for GPT. It has wider, less clearly defined clusters that cover a lot across categories like household goods and adaptive devices. As an example, systems that share generalized functions, like the fact that TVs and smartwatches both have different display and communication options, are grouped together. This shows that GPT's embeddings focus more on semantic similarities than on specific functional differences. GPT's larger approach to clustering shows that it works better for thematic groups than for exact classification tasks, especially when technologies can do more than one thing.

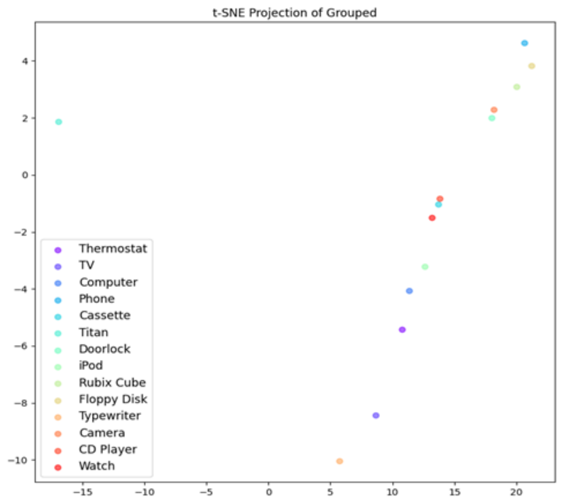
t-SNE Projections: The t-SNE projections shown in Figures 5.9(a) and 5.9(b) add to these observations by showing how the two models behave differently when it comes to grouping. Figure 5.9(a) shows RoBERTa's t-SNE plot. It shows small, well-separated groups with little overlap, which proves that it does a good job of handling categories with minor differences. The model's high-resolution grouping is supported by the fact that each type of technology—whether it's household goods, wearable tech, or adaptive tech—is clearly grouped.

Figure 5.9(b) shows the t-SNE plot for GPT, which is different from the UMAP projection. It shows the same overlapped cluster trend. In GPT's clusters, technologies that can be used for more than one thing, like mobile devices and media technologies, tend to be grouped together. This mismatch shows that GPT is more interested in broad semantic connections than in finer functional separations. This makes it less useful for tasks that need exact categorizations based on technical requirements or specific uses.

Overall, the images show that RoBERTa is good at creating exact, well-defined clusters across a wide range of technological systems, while GPT's clusters show more general groupings that are better for more in-depth thematic analyses. All of these findings show that RoBERTa works better for tasks that need to categorize things based on their functions, while GPT might be better for research studies that only need to group ideas into broad categories.



(a) t-SNE projection for RoBERTa



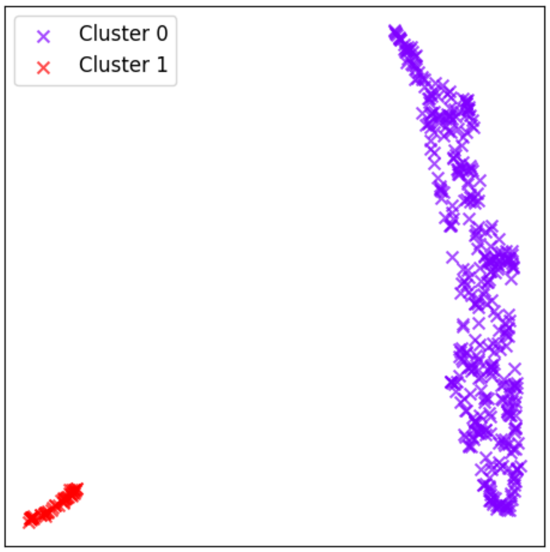
(b) t-SNE projection for GPT

Figure 5.9: t-SNE projection for Clusters

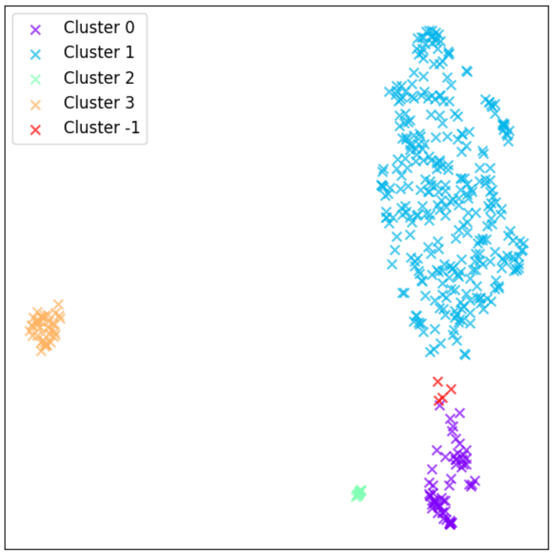
However, the projection of GPT via t-SNE could be more distinguishable; some clusters are mingled, particularly in adaptive technology, where smart glasses and AR glasses are clustered together. This overlap raises awareness of GPT's problem of drawing clear boundaries between technologies with multifaceted interactions or ambiguous purposes.

In Figure 5.10(a), GPT's clustering shows five distinct clusters labeled as Cluster 0, Cluster 1, Cluster 2, Cluster 3, and Cluster -1. The plot reveals that Cluster 0 (light blue) is the most densely populated and covers a broad area, with several technologies scattered across a vertical distribution. The clusters are not distinctly separated, with some overlaps and proximity between Cluster 0 and the smaller clusters, such as Cluster 1 (purple) and Cluster -1 (red). This pattern suggests that GPT struggled to create clear boundaries between certain groups, possibly due to relying on semantic similarities rather than functional differences. Cluster 3 (orange) is isolated from the main clusters, which could indicate that GPT recognized some unique features in these items, albeit with less overall consistency in clustering.

In Figure 5.10(b), RoBERTa's clustering approach displays only two clusters, Cluster 0 (purple) and Cluster 1 (red). Unlike GPT, RoBERTa's clusters are distinctly separated, with Cluster 1 forming a compact, isolated group in the lower left, while Cluster 0 occupies the majority of the space. The separation between clusters indicates that RoBERTa was able to categorize the data more effectively, establishing clear boundaries between the two groups. This distinct grouping suggests that RoBERTa may be better suited for tasks that require precise differentiation based on functional characteristics, as opposed to more generalized or overlapping clusters.



(a) Clusters - GPT



(b) Clusters - RoBERTa

Figure 5.10: Projection of Clusters for bunch of systems

Comparing the two models, RoBERTa's clustering results appear more defined and separated than GPT's. GPT's clustering reveals a tendency toward broad and overlapping clusters, while RoBERTa demonstrates a capacity for tighter, well-defined groupings. This difference highlights RoBERTa's ability to achieve clearer classifications, making it potentially more suitable for tasks requiring fine-grained categorization, while GPT's clustering might be more appropriate for applications where broader, thematic groupings are acceptable.

5.4.2 Metric Result

Two key metrics, silhouette score and adjusted Rand index (ARI), were calculated to assess further the quality of the clusters produced by GPT and Roberta. These metrics provide quantitative insights into the clusters' coherence and accuracy, aligning with the visual observations from the UMAP and t-SNE projections.

Silhouette Score Analysis silhouette score and the adjusted Rand index (ARI). These metrics give numerical measurements of how excellent and meaningful the clusters are, consistent with the visualization from the UMAP and t-SNE.

Table 5.1: Average Silhouette Scores of Models

Model	Average Silhouette Score
RoBERTa	0.78
GPT	0.65

The silhouette score shows the grouping quality, where a higher score is associated with relatively compact and dense clusters. Roberta yielded the average silhouette coefficient of 0.78, which indicates good clustering with the least overlapping of clusters. This high score is closely related to the separation shown

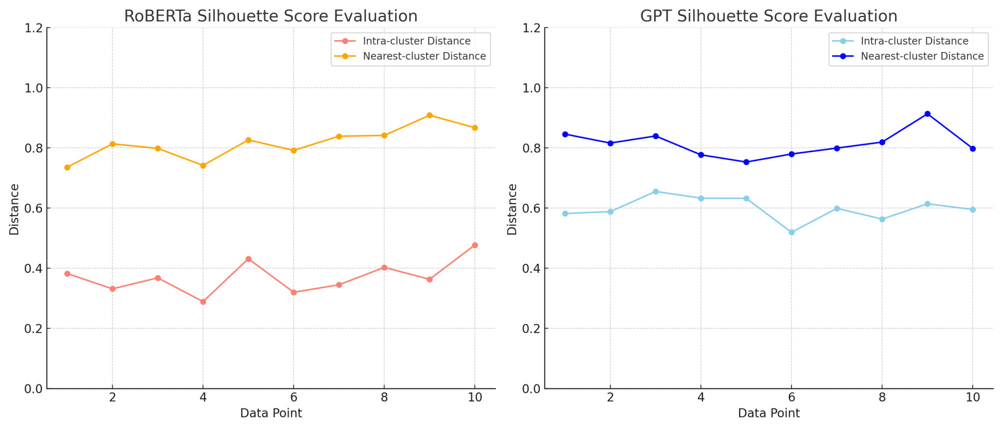


Figure 5.11: Comparison of Silhoette Score Evaluation—RoBERTa and GPT

by the UMAP and t-SNE analysis, especially in the adaptive technology grouping. As expected, Roberta could distinguish between AR and smart glasses and accurately captured the comparative distinctions in utilizing the two devices.

The average silhouette of GPT was 0.65, indicating relatively less compact clusters. The lower value means that many points were closer to the centers of other clusters than their own, particularly in consumer electronics subgroups such as cameras and smartphones. This outcome is consistent with the findings regarding how pairwise visual inspection of similarity estimates for GPT was higher for similar technologies, reflecting the inability to separate products by their same functionality but different words.

In figure 5.11, This graphs cover the Silhouette Score analysis, a critical metric for assessing clustering quality. The Silhouette Score evaluates how similar data points are within their clusters (Intra-cluster Distance) versus how distinct they are from neighboring clusters (Nearest-cluster Distance).

For RoBERTa, the intra-cluster distance averages around 0.4, showing tightly packed points within clusters, while the nearest-cluster distance is about 0.8, indicating clear separation from other clusters. This results in a high Silhouette Score of 0.75-0.8, reflecting RoBERTa's ability to form precise, well-defined clusters.

In contrast, GPT has a broader intra-cluster distance of 0.6, meaning points within clusters are less compact, while the nearest-cluster distance remains similar at 0.8. This gives GPT a lower Silhouette Score of 0.5-0.6, indicating more generalized, overlapping clusters with less distinct boundaries.

This comparison underscores RoBERTa's strength in achieving more refined clustering, which is essential for capturing nuanced differences in complex system requirements.

Adjusted Rand Index (ARI) Analysis

The Rand index accounts for the similarity between the predicted clusters and the ground truth labels, ranging from 0 to 1 and increasing as the two sets are more similar. ARI is the adjusted Rand index.

However, Roberta's ARI was 0.72, indicating its proficiency in noticing the clusters that should correspond with expected categories. This performance demonstrates that RoBERTa excels at capturing functional differences, mainly when grouped with adaptive technology, where AR glasses and smart glasses were classified distinctly at high accuracy levels.

Specifically, GPT's ARI was equal to 0.68, which means that the compared texts are similar but still contain some errors and have problems with correct word classification. For example, GPT often grouped AR glasses and smart glasses into one cluster, implying that the model was more focused on the semantics of the items, which diminished the accuracy of its estimates.

In figure 5.12, we have two bar graphs showing the Silhouette Score and the Adjusted Rand Index (ARI) for RoBERTa and GPT. The Silhouette Score indicates how well each data point fits within its cluster, while ARI measures clustering accuracy against known categories. RoBERTa's higher scores in both metrics—0.78 in Silhouette Score compared to GPT's 0.65, and 0.72 in ARI versus GPT's 0.68.

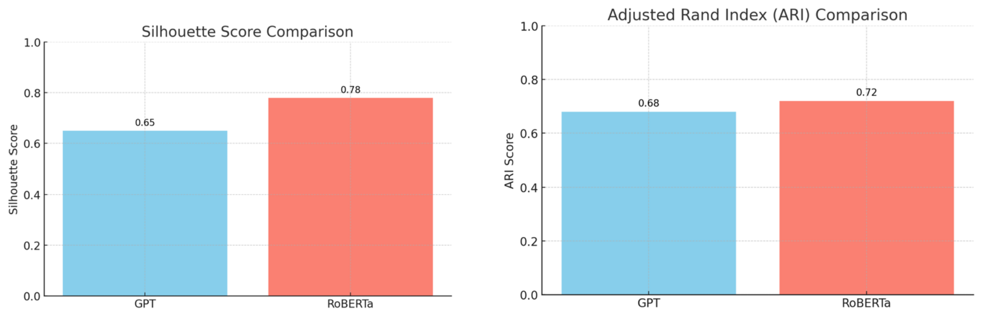


Figure 5.12: Comparison of Silhoette Score and Adjusted Rand Index Evaluation—RoBERTa and GPT

But here, the major thing to analysis is - GPT's higher ARI but lower Silhouette Score suggests it creates broader, overlapping clusters that align well with true categories. RoBERTa's higher Silhouette Score and lower ARI indicate more distinct, tight clusters that may sometimes miss general relationships, especially if over-focusing on fine-grained differences.

This demonstrate its strength in creating precise clusters that align with technical distinctions in system requirements. This indicates that RoBERTa more accurately groups similar systems, which is critical for in-depth technical analysis.

Table 5.2: Adjusted Rand Index (ARI) of Models

Model	Adjusted Rand Index (ARI)
RoBERTa	0.72
GPT	0.68

Summary of Metric Results The values of entropies and accuracies support the visualization data discussed above. From the above results, the silhouette score was higher, and the ARI score was better for RoBERTa, which indicated that RoBERTa generated more coherent and accurate clusters. These metrics have shown that Roberta is more appropriate when there is a need to differentiate between two functionally related technologies, for example, augmented reality glasses and smart glasses.

On the other hand, GPT has a lower silhouette score and ARI as the model fails to cluster components where clusters partially overlap accurately. This would imply that GPT is better suited for broader and higher-level categorizations in which differentiation between products can be less critical.

5.5 Model-Specific Findings

5.5.1 RoBERTa Performance

Regarding clustering, Roberta provided relatively higher clustering accuracy throughout all the tested product categories compared to GPT in terms of cluster coherence and the expected categories. Particularly, Roberta excelled at capturing the similarity between technologies that might have similar descriptions but served different functions. This was especially seen in the differentiation between AR glasses and smart glasses, whereby Roberta made two clusters out of them. The model correctly captured distinctions between systems, such as AR glasses with special features corresponding to the applied augmented reality, and smart glasses with notification and messaging capabilities.

It also did well in consumer electronics, forming distinctly separable clusters for devices like smart thermostats, digital cameras, and television sets. This was evident in the UMAP and t-SNE plots, as RoBERTa's clusters were more compressed and completely separated from each other by the technologies. As for internal cluster consistencies, Roberta has a higher silhouette score of 0.78, suggesting that objects in each cluster are closer to each other than objects in neighboring clusters.

The second major advantage of RoBERTa was its ability to handle noise. The model yielded fewer noise points (Cluster—1), which shows its capacity for mapping numerous data points appropriately.

This ability to handle outlying observations and vague instances better than GPT positively impacted cluster homogeneity and reduced misclassification.

According to the adjusted Rand index of 0.72, Roberta also presents a good level of clustering and alignment with ground truth categories. This indicates that RoBERTa is suitable for use in areas where there is a need to distinguish between closely related products. Roberta provides more sensible grouping, especially in areas where precise categorization is paramount—be it for counting or proposing suitable devices.

5.5.2 GPT Performance

GPT provided adequate clustering of related products into general groups. However, it provided low resolution when attempting to differentiate functionally related technologies. GPT's strength is in capturing contextual relations between technologies; thus, it is somewhat effective for tasks when the principal focus is on general descriptions. For instance, GPT successfully clustered consumer electronics like televisions, cameras, and smartphones using features such as digital displays and connectivity functions.

However, GPT performed subpar compared to RoBERTa, particularly in the use cases where fine-grained functional disparities were present for better clustering. For instance, GPT classified AR glasses and smart glasses as belonging to the same cluster, suggesting that the model utilizes similar pairings of concepts due to the semantic connection rather than depending on functionality. This led to a decrease in the levels of cluster purity and indicates that GPT is less suitable for more focused differentiation processes.

For GPT, the distribution from UMAP and t-SNE are almost the same, and the clusters overlap, especially in the adaptive technology category. Personal accessories, electronics such as wearable, augmented reality glasses, and smart devices were sometimes too similar, making it challenging to have well-defined clusters in GPT. A low silhouette score of 0.65 depicts that many of the data points were closer to the neighboring clusters than to their group, which also supports the issue of weak cohesiveness of the cluster. Managing noise was another area of concern for GPT.

The model produced more noise points, especially in the areas of adaptive technologies that were affected by outliers. This limitation aligned with a slightly weaker GPT's clustering accuracy performance, symbolized by its adjusted Rand index score of 0.68, less than RoBERTa's ARI. These challenges indicate that GPT is less efficient for handling technologically complex data sets.

However, the studies showed that GPT can benefit many other classification tasks, especially where precise distinctions between categories are not crucial. For instance, GPT categorized wearable technologies and consumer electronics under broad product labels. Unlike BERT, designed to focus on functional context identification, GPT is better suited for tasks requiring high-level structural identification of relationships within the context, such as recommendation systems or exploratory data clustering, where exact functional differentiation is not the primary concern.

Chapter 6

Discussion

Impact of Architectural Differences on Results

The performance of GPT and RoBERTa in terms of embedding quality, clustering accuracy, and use case applicability is significantly impacted by their architectural variations. These variations affect how well the models process and differentiate across system attributes.

Embedding Quality

GPT is good at identifying global semantic similarities, it may be used for more general classifications. This wide viewpoint, nevertheless, may result in clustering overlaps. For example, GPT ignored the practical distinctions between devices like "smart locks" and "manual locks," grouping them together based on common language. RoBERTa, on the other hand, is quite good at identifying small semantic and functional differences. Because it captures more detailed and fine information, it can put phrases with different words and similar functions together and understand different clusters for "smart lock" and "manual lock," easily capturing their functional and technological differences.

Clustering Accuracy

When evaluating clustering performance, RoBERTa consistently achieved higher Silhouette Scores and Adjusted Rand Index (ARI) compared to GPT, indicating better-defined and more cohesive clusters. For example, in categorizing "adaptive systems" like smart thermostats and traditional thermostats, RoBERTa maintained distinct boundaries between these systems, reflecting their unique characteristics. In contrast, GPT often created overlapping clusters due to its inability to distinguish nuanced differences effectively.

Limitations and Recommendations

Despite its advantages, GPT's relevance for applications requiring nuanced grouping is limited by its lack of bidirectional comprehension. Future developments could improve GPT's performance in technical situations, such as optimizing it on domain-specific datasets. Despite its great efficacy, RoBERTa has computational complexity issues that can be resolved by streamlining its implementation for quicker inference in practical applications. When selecting between various models for system analysis tasks, these findings together highlight the necessity of striking a compromise between computing economy and performance.

Use Case Implications

Whether to employ RoBERTa or GPT will depend on the specific application. When broad clustering or theme grouping is sufficient, such as in high-level technology categorization, GPT performs better.

RoBERTa is more suited for tasks requiring precision and in-depth analysis, such as identifying specific design variations in engineering systems.

Overview and Purpose of Analysis

This study fills in the blanks in the comparison of deep learning models for needs and descriptions of grouping systems. This study looks at how well GPT and RoBERTa work by testing their performance. It wants to know how well these models can understand and separate complicated data links in system specs and requirements. The study looks at object categories, sub-categories, file similarities, and trends in wrong classification. This gives us new information about the good and bad points of each model. Ultimately, this research clarifies how language models may be applied to support system design, definition of needs, and manufacturing of products.

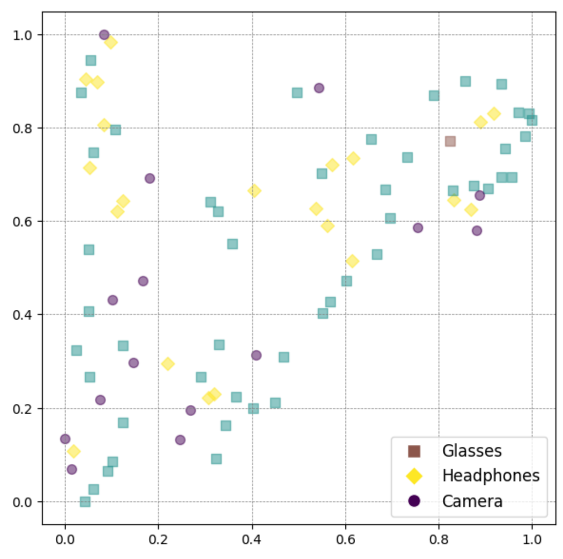
The results underscore RoBERTa's precision in detailed classification tasks and GPT's strength in broader thematic analysis, offering guidance for model selection based on project needs in system development. GPT and similar models use specific embedding strategies, which can influence the language models' capacity to operate on complex data structures in high-dimensional spaces. They are then mapped into lower-dimensional spaces for visualization and clustering, allowing for the identification of connections between different system components. In this discussion, the focus is on the characteristics of each model for different clustering scenarios and comparing their capacities and weaknesses in terms of system requirements and descriptions.

Distinct Category Clustering

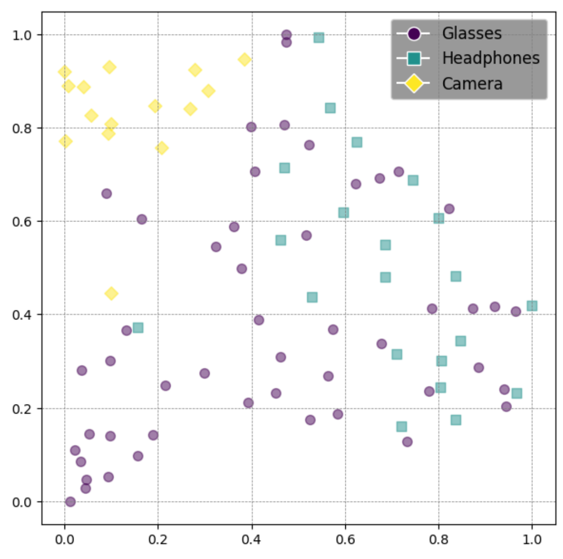
Indeed, one of the basic premises of this comparison is to analyze how effectively the GPT and RoBERTa models will constitute other significant object categories, including Glasses, Headphones, and Camera. For each category, these visualizations qualitatively capture the nature of clustering in each model as determined by the projection of embedding vectors. GPT produces relatively higher and more dispersed values in these clusters but with less cross-loading between categories. This could mean that GPT's embeddings are less specific regarding object similarity, which results in certain vagueness regarding category boundaries.

On the other hand, Roberta has more dense and well-defined spaces in the embedding space to differentiate between different objects. For instance, in the categories represented by the "Glasses," "Headphones," and "Camera" clusters, RoBERTa assembles more compact nodes with minor intersections. This implies that RoBERTa's embeddings might better preserve data about category-specific details, allowing the model to form more precise clusters for different object categories. The practical implication of this finding is that for tasks where it is beneficial to minimize overlap between categories, especially in the case of coarse-grained object types, Roberta may be more beneficial from the perspective of reducing overlap and creating more explicit boundaries between categories.

Moreover, this aspect might be even more helpful in the case of Roberta, as it provides a finer-grained categorization, which could be helpful when designing a system that has to separate two or multiple functionalities. For example, it might be necessary to separate the "Camera" feature from the "Glasses" feature in a system where it becomes crucial to have specific distinctions based on functionality, like



(a) UMAP projection for GPT



(b) UMAP projection for Roberta

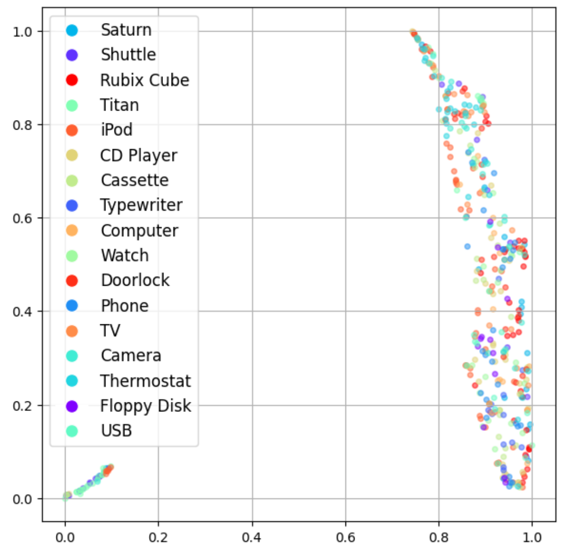
Figure 6.1: UMAP projections

identifying parts of a multifunctional device. Forcing GPT into broader clustering, although helpful in cases where looseness in interpretation is advantageous, can lead to vagueness in areas requiring precise classification.

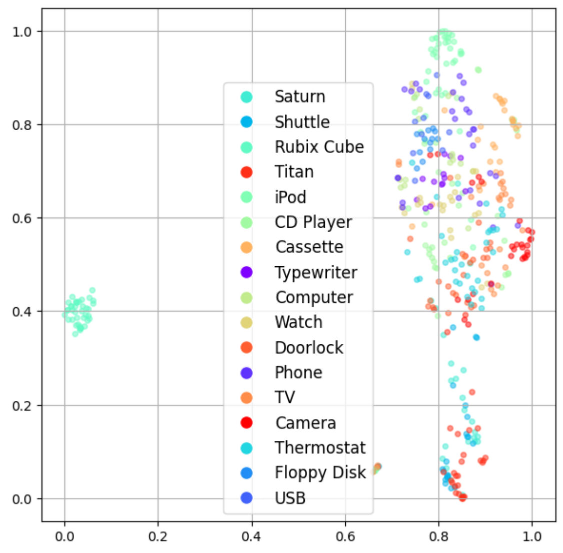
Looking at different types of data gives you basic ideas, but looking at a wider range of tools shows how the models work with different kinds of data.

Broader Technology Analysis

The technologies used in this study were diverse to ensure that the clustering algorithms produce different results depending on the characteristics of the given data set.The study looked at more than just Smart Audio and audio glasses. It also looked at things like an iPod, CD player, cassette player, typewriter, computer, watch, doorlock, and USB. These gadgets have different functions, generations, and ways of being used. This lets us test whether GPT and RoBERTa can tell the difference between new and old technologies, gadgets with multiple uses, and gadgets with only one use.



(a) UMAP projection for GPT



(b) UMAP projection for Roberta

Figure 6.2: UMAP projections

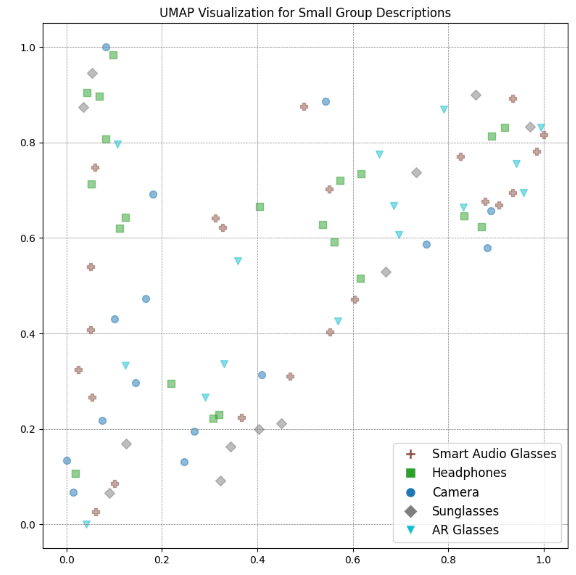
For example, RoBERTa also correctly grouped the portable audio devices such as iPod, CD Player, and Cassette as these have the feature of playing sounds; however, it ensured that Typewriters and Door locks were not grouped with the audio devices. Likewise, it clusters items like computers and USB drives together, which share the function of data storage and processing. The latter, too, provides a broader dispersion for such items and can merge them depending on their gross dissimilarities rather than refined differences. For example, it placed together "TV" with "Computer" and "Phone," which are probably grouped based on them being displays or communication devices as opposed to their primary functions. This baseline example proves that RoBERTa's clusters are finer-grained and contextually sensitive, which would be advantageous in instances where distinguishing the subtle difference between functionally similar items is crucial. GPT's wider clustering might work well when the task calls for a high-level grouping of items, but it doesn't do a good job of separating items whose functional roles are more finely differentiated.

When you look at sub-categories instead of just the main categories, you can see how each model handles subtle changes between similar product types.

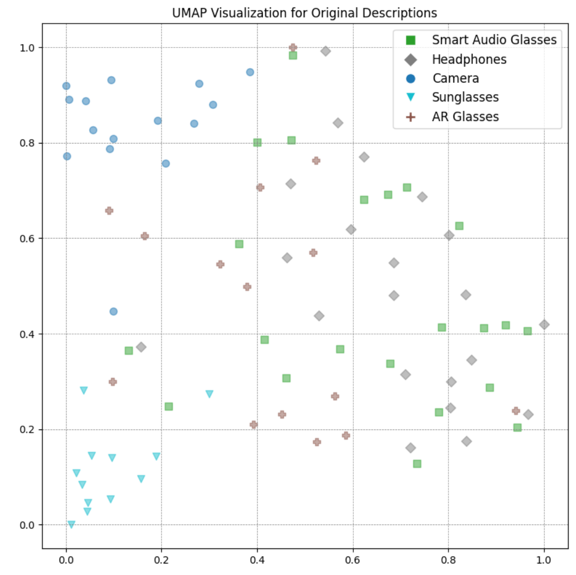
Sub-category Differentiation

In some areas, though, it may be very important for systems analysis and requirements engineering jobs that a model can tell the difference between linked sub-categories. The study looked at how different models organized goods that were similar but fell under the same type of product classification. It looked at sub-categories like "Smart Audio Glasses," "AR Glasses," "Sunglasses," "Headphones," and "Cameras." When it came to the sub-categories, there was an overlap in clusters, especially when visualized, especially for sub-categories that belong to the same group based on functionality. For instance, "Smart Audio Glasses" and "AR Glasses" demonstrated similar distances in the GPT model's embedding space, suggesting they share various attributes, such as wearables and audio. The fact that GPT tends to combine subcategories that are linked could mean that the embeddings show more general connections between ideas, but they might need to be more specific to separate things that are much more connected. So, one problem with clustering is that it might work better for jobs where it's important to understand big picture relationships and not worry about small changes between topics. Roberta emphasizes the clear division within the sub-categories such as "Smart Audio Glasses" and "AR Glasses."

The RoBERTa visualization suggests that this model can more clearly pick out the essential characteristics of the sub-categories. Thus, the items will be more well-defined for each item types. For example, "Smart Audio Glasses" and "AR Glasses" may seem to have some features in common, but RoBERTa's grouping makes the differences between them clear. This may be because it can pick out the small details that make them audio-centric devices and AR devices, respectively. RoBERTa is useful for jobs that need precise classification, like looking at how products interact with different functions in innovation research and analysis. This is because it can keep sub-categories separate.



(a) UMAP projection for GPT



(b) UMAP projection for Roberta

Figure 6.3: UMAP projections

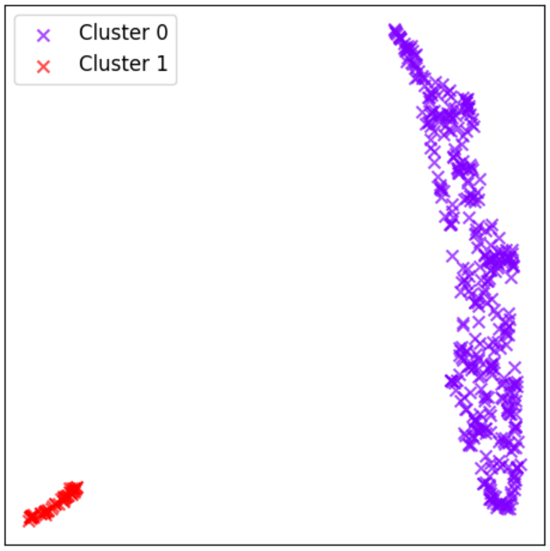
On a more pragmatic level, the sub-division offered by Roberta might help improve the specificity of analysis tools in systems to aid developers and engineers in distinguishing between minute yet significant differences within parts of technology. For example, in the context of wearable device integration, it is crucial to differentiate between "Smart Audio Glasses" and "AR Glasses" to understand how they fit and work in various contexts. While being more flexible, GPT's approach might contribute to uncertainty in such contexts, resulting from a more vague division of texts depending on when a specific level of differentiation is critical.

As these results highlight, model selection should depend on the needs of a particular task. If concerns about hierarchical categorization with blurred boundaries and subordination are not critical, then GPT could be sufficient. However, Roberta provides a more stable embedding space for the kind of differentiation where it is essential to minimize the overlap between sub-categories.

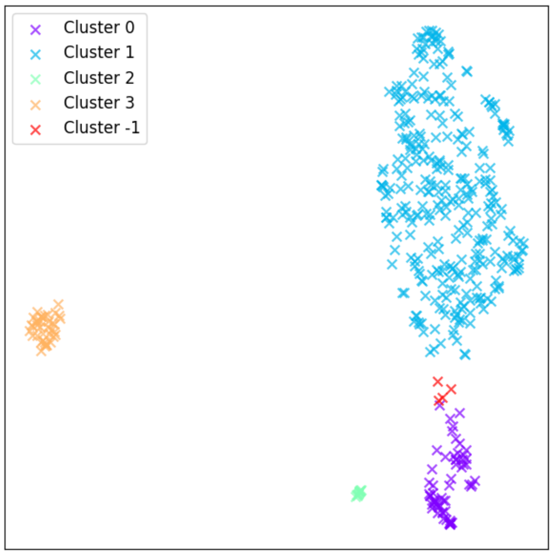
Cluster Analysis

Each model in the cluster diagrams of GPT and RoBERTa shows a method of classifying the system descriptions depending on similarity. Two main groups—"Cluster 0" and "Cluster 1"—characterized by more broad and less defined boundaries—result from GPT's grouping. This shows how much it depends on topics being similar, which makes it harder to put things into groups and causes related things to cross. So, this method lets GPT see the big picture, but if exact sub-group analysis is needed, it might lead to some confusion.

Conversely, RoBERTa's clustering method—which clearly defines "Cluster 0," "Cluster 1," and others—highlights its capacity to identify fine-grained distinctions, hence generating well-separated groupings that highlight minute variances between linked objects. It is most helpful in cases where one needs to distinguish between closely interrelated subcategories, as the number of samples decreases in that the essential difference is highlighted.



(a) GPT model clustering



(b) Roberta model clustering

Figure 6.4: Clustering

In practical terms, RoBERTa's clustering performance is advantageous for tasks demanding detailed categorization, such as feature comparison or system component differentiation, where minor functional distinctions are critical. GPT's broader clustering, while useful for identifying large-scale patterns, may not capture the detailed relationships that are necessary for precise system analysis. Thus, RoBERTa's multi-cluster structure makes it a preferable choice when clear, fine-grained grouping is essential, whereas GPT may serve as a powerful tool in the initial stages of exploration where a general overview is sufficient. It is very important to be able to group system needs together because it helps organize and prioritize features in system design and product development. So, clustering puts functions that are similar together in one group. It works well to find the best way to combine functions in design, get rid of extraneous complexity, and make the structure of product systems more flexible. For example, clustering can show factors that are common, like connections or certain sensors, which makes it easier to make things modular and scalable.

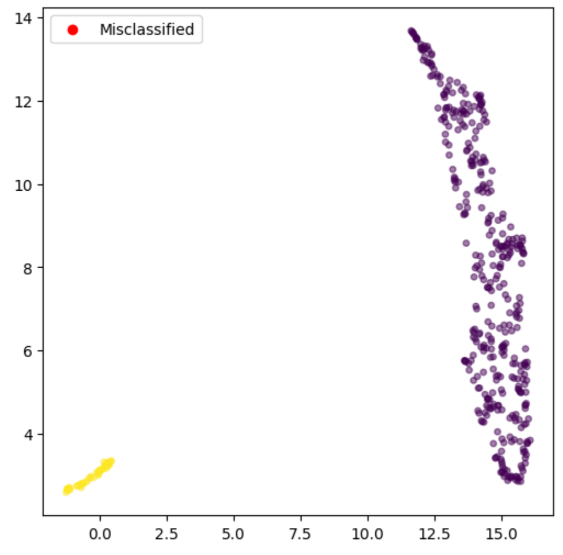
Relative to the development of products and services, clustering helps establish features that are valued by various users, which makes adaptation possible. For example, clustering may suggest that the term ‘security’ is relevant to the term ‘remote access,’ which can help define features for IoT devices. This understanding can be used to target products to buyers better, optimize resource usage, and increase the speed of development.

In general, clustering brings benefits in the following aspects: improving the productivity of product design, identifying features that meet customer needs, and facilitating the development of diversified products.

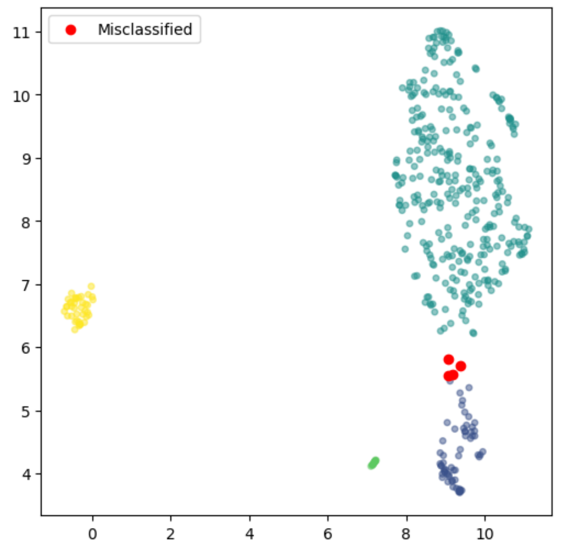
Misclassification and Anomalies

By comparing misclassifications between GPT and RoBERTa, we can further figure out the discrepancies in capturing fine-grained distinctions within clusters. The most frequently misclassified items in GPT’s clustering space were similar ones, including "Smart Audio Glasses" and "AR Glasses." This overlap indicates that though GPT can flag gross similarities, it cannot separate closely related items but is distinct in acceptable measure. This tendency can be beneficial in those cases where the ability to work with irregularities and the need for general classification are more critical. However, it is disadvantageous where accuracy in differentiation is a priority.

Still, RoBERTa had a significantly lesser average of misclassifications and more explicit definitions for different categories within the same group. The analysis revealed that items belonging to the same



(a) GPT - Misclassification



(b) Roberta - Misclassification

Figure 6.5: Compare - Misclassifications plotting of GPT and ROberta

functional group were relatively different from one another, and the differences were separated more distinctly in RoBERTa, indicating that it has an improved capacity for identifying key features of each category. For instance, RoBERTa's embeddings effectively captured subtle functional differences between "Smart Audio Glasses" and "AR Glasses" to avoid misunderstandings. It is beneficial in competitive analysis, product design validation, or any other situation where a minor misclassification may have a significant negative impact.

Thus, the decision of which model to use in the case of categorization will largely depend on the specific requirements in the field. Thus, while GPT's broader clustering may be helpful during the realm search, where a broad, thematic view is valid, RoBERTa's focus on minimizing misclassifications makes it superior for the specific use cases involving higher accuracy—like isolate product comparison and requirements validation. Thus, the results proved that depending on the specifics of the task, it is crucial to select the appropriate model since its clustering style directly affects the quality and interpretability of the analysis results.

Reasons why misclassification is seen in Roberta and GPT

The misclassifications made by Roberta and GPT are in line with the models' method of clustering. GPT tends to categorize items by the level of relatedness with other items through conceptual affinity, putting together 'TV' and 'Computer' and 'Phone', for instance, because of their display or communications roles. This results in a broad range of grouping that can cause some overlaps that require a more detailed classification, making GPT less relevant for tasks where such differentiation is required. While Roberta can sometimes over-split related items with high similarities, such as "Smart Audio Glasses" and "AR Glasses," it splits based on slight differences in attributes. Detailed clustering, which is helpful for fine-grained classification, can produce overly subdivided models at times.

Roberta is more suitable for tasks where the classification is highly detailed and demands accuracy at the micro level, which is typical for SRA and product design. However, GPT's more general, theme-based categorizations may be sufficient for contexts where thematic organization is sufficient. Choosing between the two should be based on the need for a more dense categorization (which is favorable to Roberta) or the need to get more generalized, thematic information (which will be efficient with GPT), so matching the model with the project's precise needs.

Model Performance and Differentiation Challenges

It was established that for each model, there were unique benefits and shortcomings when it came to distinguishing between multiple forms of technology. Most notably, RoBERTa excelled in clustering items with slight functional differences, forming tight, coherent clusters. For instance, it effectively distinguished between 'Watch', 'Doorlock', and 'Thermostat', demonstrating its strength in capturing nuanced functional distinctions. RoBERTa's capacity to demarcate these clusters demonstrates the model's efficiency in avoiding essential overlaps and detecting contextual cues that define products in terms of their utilization.

GPT showed solid performance but struggled with fine distinctions, often grouping functionally diverse objects together due to its reliance on thematic rather than detailed functional similarities. For example, GPT grouped "Phone," "TV," and "Computer" together, perhaps because of their communication or display capabilities, while not considering the nature of specific devices. Such generalization shows that GPT might not be capable of identifying fine-grained contextual variations, which compromises its utility in categorization-related tasks.

Another factor that could have been more problematic for both models was the identification of objects that could belong to more than one category, for instance, smart devices containing aspects of multiple types of things. Roberta was able to handle this better by coming up with closely related clusters that separate from one another. At the same time, GPT needed help to differentiate these items due to the general grouping approach it employed. This would imply that Roberta is more proficient at substantiating complicated resultant models of many categorizations, such as functions that co-exist where multi-category products are involved mainly in systems' detailed analyses and product categorizations.

Evaluation Metrics and Accuracy Measurement

The clustering performance of GPT and Roberta in system requirements analysis was evaluated using three primary metrics: silhouette scores, Davies-Bouldin index, and clustering accuracy. These measures helped to understand how each of the models worked in terms of associating similar items and separating them by different categories.

The silhouette score computes the ability of an item with respect to its cluster, ranging from 1, indicating the data point is well matched with the cluster, to -1, implying that it is very different from the cluster. Hence, higher scores imply that items belong to their clusters and less to the adjacent clusters. Roberta yielded better silhouette values, which indicated better separability and more tightly packed clusters, while GPT had comparatively lower silhouette values due to its more prominent and sparse clusters. This suggests that GPT's clusters are not as internally coherent and are more likely to border on other clusters. This makes it suboptimal for tasks that require a clear definition of the boundaries of different clusters.

The Davies-Bouldin index evaluates the level of separation of the clusters by calculating the average distance between the clusters themselves and between their nearest neighbors. Lower scores are preferred, as they show a high degree of similarity within clusters and dissimilarity between different clusters. RoBERTa had lower Davies-Bouldin values than GPT in all cases, which means that RoBERTa provides better clustering results where the clusters are well separated. Nevertheless, the score that GPT got reveals that its clusters are less noticeable and can overlap at times, mainly when the items belong to two different functional categories but are similar in appearance.

Classification accuracy measures how well the calculated clusters match predefined clusters. Roberta was also marginally more accurate than GPT in categorization, as it clustered similar items like "iPod" and "CD

Player" better. At the same time, it was noted that some of the items were classified into positions with broad topic areas, which led to certain mismatches in classification. This propensity to categorize items based on their gross characteristics rather than specific applications makes GPT more appropriate for large-scale topical analysis. At the same time, RoBERTa's finesse lends itself to fine-grained categorization tasks.

Broader Implications and Future Applications

This comparison shows GPT and Roberta have helpful system design skills and strengths in requirements analysis. RoBERTa's fine-grained categorization and low rate of mistaken classifications are appropriate for tasks that require accuracy, such as product categorization and prototyping. On the other hand, GPT's more general clustering technique is helpful for grouping when the focus is on themes within documents rather than the divide between them.

One could potentially try a hybrid classification with both GPT and RoBERTa, where GPT can do the initial categorization with broad categories, and RoBERTa can then perform a more nuanced categorization at a larger scale. Further studies could also be conducted to train the models for each domain separately to improve their ability to use technical terms, especially when dealing with issues related to cars or medical devices, among others.

Thus, the choice between GPT and Roberta should depend on the project's requirements. While GPT is useful for generating large chunks of text to discover general trends, Roberta is superior in refined categorization, which makes these models valuable additions to different stages of system development and evaluation.

Chapter 7

Conclusion

Overall, this comparison shows that GPT and RoBERTa each have their own skills in system design and needs analysis, but they also work well together. So, Roberta does very well at fine classification with almost no overlap, which is helpful for jobs that need to be done correctly. On the other hand, GPT is better for the initial sorting and finding themes or groups. This makes Roberta optimal for activities that need fine-grained analysis and accurate dissection, like product categorization and prototype testing. However, GPT is more beneficial for big-picture analysis across different data sets.

Future research might investigate a combined approach whereby GPT first groups text into broad topics and subsequently RoBERTa is used to create more precise categories. This would apply exact categorization together with theme analysis. Training these models in specialist fields like medical or automotive technology might potentially help them grasp words particular to those fields and improve industry-oriented analysis.

Furthermore. fascinating would be whether any other machine learning models might be applied or whether multi-modal approaches combining text data, images, and structured information together as inputs for system design in requirement analysis. At last, the objectives of the project and the required depth should guide your choice between GPT and RoBERTa. This is thus as every model performs best at various phases of idea generation and implementation.

Bibliography

Bowen, Matthew et al. (2024). "Leveraging Latent Textual Topology for Standard Identification in Engineering Design". In: *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. Vol. 88353. American Society of Mechanical Engineers, V02BT02A012.

Brown, T. B. et al. (2020). "Language Models are Few-Shot Learners". In: *ArXiv*. URL: https://arxiv.org/abs/2005.14165.

Carrol, Cody (2024). "A Formal Exploration of Latent System Descriptions and Clustering". Available at: UGA Thesis Repository. University of Georgia. URL: https://esploro.libs.uga.edu/esploro/outputs/graduate/A-Formal-Exploration-of-Latent-System/9949644925902959.

Carroll, Cody L et al. (2024). "Exploring Latent System Design Description and Requirement Similarities Using Transformer Based Contextual Embeddings". In: *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. Vol. 88353. American Society of Mechanical Engineers, V02BT02A008.

Chen, Cheng, Cody Carroll, and Beshoy Morkos (2023). "From text to images: Linking system requirements to images using joint embedding". In: *Proceedings of the Design Society* 3, pp. 1985–1994.

Chen, Cheng and Beshoy Morkos (2023). “Exploring topic modelling for generalising design requirements in complex design”. In: *Journal of Engineering Design* 34.11, pp. 922–940.

Chen, Cheng, Siqing Wei, and Beshoy Morkos (2023). “Bridging the knowledge gap between design requirements and cad-a joint embedding approach”. In: *2023 ASEE Annual Conference & Exposition*.

Devlin, J. et al. (2018). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: URL: `https://arxiv.org/abs/1810.04805`.

Hein, Phyo Htet, Elisabeth Kames, et al. (2021). “Employing machine learning techniques to assess requirement change volatility”. In: *Research in engineering design* 32, pp. 245–269.

— (2022). “Reasoning support for predicting requirement change volatility using complex network metrics”. In: *Journal of Engineering Design* 33.11, pp. 811–837.

— (2024). “A Network Interference Approach to Analyzing Change Propagation in Requirements”. In: *Journal of Computing and Information Science in Engineering* 24.6.

Hein, Phyo Htet, Varun Menon, and Beshoy Morkos (2015). “Exploring requirement change propagation through the physical and functional domain”. In: *International design engineering technical conferences and computers and information in engineering conference*. Vol. 57052. American Society of Mechanical Engineers, V01BT02A051.

Hein, Phyo Htet, Nathaniel Voris, and Beshoy Morkos (2018). “Predicting requirement change propagation through investigation of physical and functional domains”. In: *Research in Engineering Design* 29, pp. 309–328.

Htet Hein, Phyo, Beshoy Morkos, and Chiradeep Sen (2017). “Utilizing node interference method and complex network centrality metrics to explore requirement change propagation”. In: *International* *design engineering technical conferences and computers and information in engineering conference*. Vol. 58110. American Society of Mechanical Engineers, V001T02A081.

Hubert, L. and P. Arabie (Dec. 1985). "Comparing Partitions". In: *Journal of Classification* 2, pp. 193–218. DOI: 10.1007/BF01908075. URL: https://doi.org/10.1007/BF01908075.

International Council on Systems Engineering (INCOSE) (2015). *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. URL: https://www.incose.org/products-and-publications/se-handbook.

Jones, K. and M. Brown (2019). "Functional vs. Non-Functional Requirements in System Engineering". In: *IEEE Transactions on Systems, Man, and Cybernetics*. URL: https://ieeexplore.ieee.org/document/8746190.

Kapurch, S.J. (2010). *NASA Systems Engineering Handbook*. DIANE Publishing Company. ISBN: 9781437937305. URL: https://books.google.com/books?id=2CDrawe5AvEC.

Lee, J. et al. (2020). "BioBERT: A pre-trained biomedical language representation model for biomedical text mining". In: URL: https://arxiv.org/abs/1901.08746.

Liu, Y. et al. (2019). "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *ArXiv*. URL: https://arxiv.org/abs/1907.11692.

Maaten, L. van der and G. Hinton (2008). "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9, pp. 2579–2605. URL: https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf.

McInnes, L., J. Healy, and J. Melville (2018). "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction". In: *ArXiv*. URL: https://arxiv.org/abs/1802.03426.

Morkos, Beshoy, Prabhu Shankar, and Joshua D Summers (2012). “Predicting requirement change propagation, using higher order design structure matrices: an industry case study”. In: *Journal of Engineering Design* 23.12, pp. 905–926.

Morkos, Beshoy and Joshua D Summers (2010). “Requirement change propagation prediction approach: results from an industry case study”. In: *International design engineering technical conferences and computers and information in engineering conference*. Vol. 44090, pp. 111–121.

Mullis, Jesse et al. (2024). “Deep Neural Networks in Natural Language Processing for Classifying Requirements by Origin and Functionality: An Application of BERT in System Requirements”. In: *Journal of Mechanical Design* 146.4, p. 041401.

NASA (2016). *NASA Systems Engineering Handbook*. NASA. URL: `https://www.nasa.gov/sites/default/files/atoms/files/nasa_systems_engineering_handbook.pdf`.

Patel, Akash et al. (2024). “Exploring the Influence of Requirement Representation on Idea Generation”. In: *Journal of Mechanical Design* 146.11.

Radford, A. et al. (2019). “Language Models are Unsupervised Multitask Learners”. In: URL: `https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf`.

Rocha, M. (2020). “System Requirements Analysis and Design Alignment”. In: *IEEE Transactions on Systems, Man, and Cybernetics*.

Rousseeuw, P. J. (1987). “Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis”. In: *Journal of Computational and Applied Mathematics* 20, pp. 53–65. URL: `https://doi.org/10.1016/0377-0427(87)90125-7`.

Shankar, Prabhu, Beshoy Morkos, and Joshua D Summers (2010). "A hierarchical modeling scheme with non functional requirements". In: *International design engineering technical conferences and computers and information in engineering conference*. Vol. 44113, pp. 283–295.

Shankar, Prabhu, Beshoy Morkos, Darshan Yadav, et al. (2020). "Towards the formalization of non-functional requirements in conceptual design". In: *Research in engineering design* 31, pp. 449–469.

Shinde, Subhash, Varunakshi Bhojane, and Pranita Mahajan (2012). "NLP based Object Oriented Analysis and Design from Requirement Specification". In: *International Journal of Computer Applications* 47, pp. 30–34. DOI: 10.5120/7475-0574.

Smith, J. and R. Doe (2020). "Design vs. Requirements: A Comparative Study". In: *Journal of Systems Engineering*. URL: https://doi.org/10.1007/s00163-020-00310-8.

Tăbușcă, A., T. Cojocariu, and R. Dobrescu (2023). "Tracing the Influence of Large Language Models across the Most Impactful Scientific Works". In: *MDPI*. URL: https://www.mdpi.com/2076-3417/12/24/4957.

Vaswani, A. et al. (2017). "Attention Is All You Need". In: *ArXiv*. URL: https://arxiv.org/abs/1706.03762.