Semantic Clustering Toolbox

Bachelor Thesis

Department of Computer Science
OST – University of Applied Sciences
Campus Rapperswil-Jona

Spring Term 2025

Author	Lukas Derungs
Advisor	Dr. Beat Tödtli, Dozent am IPM
Project Partner	IFSAR - Institute for Social Work and Social Spaces
External Co-Examiner	Leo Büttiker
Internal Co-Examiner	Prof. Dr. Markus Stolze

Abstract

This thesis presents the design, development, and evaluation of a semantic clustering toolbox intended to support non-technical users in analyzing open-ended survey responses. The motivation stems from a desire to expedite the labor- and time-intensive process of survey data analysis in research and evaluation contexts. The toolbox allows users to upload survey data, perform semantic clustering, analyze sentiment, and export results through a simplified interface. Developed using a Design Science Research methodology, it integrates embedding models for semantic representation, the K-means algorithm for clustering, dimensionality reduction for visualization, and language models for sentiment analysis. A notable feature of the system is the inclusion of cluster stability visualizations, which help users interpret the consistency of clustering outcomes across multiple runs.
The artifact was evaluated through internal clustering metrics, user feedback and requirement validation using real-world survey data provided by the IFSAR research institute. Results indicate that the toolbox effectively identifies dominant themes and supports exploratory analysis, while remaining accessible to non-technical users. Despite its utility, the toolbox has limitations, including sensitivity to input quality and the inherent subjectivity of interpreting clusters without ground truth labels. Nonetheless, the artifact fulfills its primary goal and offers a practical foundation for future enhancements and research.

Overall, this work contributes a practical and extensible tool for the semantic clustering of textual data.

Contents

List of Figures

List of Tables

List of Acronyms

IFSAR Institute for Social Work and Social Spaces

LLM Large Language Model

API Application Programming Interface

NLP Natural Language Processing

DSR Design Science Research

t-SNE t-distributed Stochastic Neighbor Embedding

MVP Minimum Viable Product

GUI Graphical User Interface

Glossary

This section explains some of the more technical terms amd concepts used in this paper.

Cluster: A group of items (e.g. survey responses) that are similar to each other.
K-means clustering: A popular machine learning algorithm that splits a dataset into a specified number of clusters (k) by minimizing the within-cluster variance.
Large Language Model (LLM): A generative artificial intelligence model trained on vast amounts of text to understand and generate human language. Examples include Google Gemini and ChatGPT.
Embedding (Vector): A list of numbers that represents data. In text analysis, an embedding captures the meaning of a word or sentence.
Embedding model: A large language model or neural network that transforms text into a numerical embedding vector.
Token: A token is a unit of text (such as a word or subword) used by language models to process and measure input length.
Application Programming Interface (API): An interface consisting of defined endpoints that can be accessed by following specific protocols. APIs enable software applications to request and use services provided by other systems.
API Key: A unique identifier used to authenticate a user and authorize access to a specific API.
Parquet file: Parquet files are a type of data storage format that organizes information to improve efficiency in saving space and speeding up data retrieval.
Artifact: A functional software tool developed to solve a specific problem and evaluated for effectiveness in Design Science Research.

Management Summary

Project Objective

The objective of this project was to develop an intuitive software toolbox that enables non-technical users to analyze open-ended survey responses. While open-ended responses often contain rich insights, they are typically difficult to process without time-consuming manual effort.

Background and Motivation

Organizations frequently use surveys to gather feedback. While multiple choice questions are easy to evaluate, open-ended responses provide more varied insight but are significantly harder to analyze. This project addresses this challenge by offering a tool that expedites the grouping of semantically similar answers.

Approach

The project followed an iterative development process, beginning with the identification of a practical problem and moving through solution design, implementation and evaluation.

The resulting toolbox allows users to:

- Upload and analyze open-ended survey responses
- Automatically group similar answers into meaningful categories
- View a visual overview of the main groupings in the data
- Understand the general sentiment of responses
- Export results for further analysis or reporting

A special focus was placed on making the tool easy to use, even for people without technical backgrounds. The Graphical User interface was kept simple while still offering valuable insights through visualizations.

A notable feature of the tool is its ability to visualize cluster stability across multiple algorithm runs. This helps users assess whether a grouping is consistent or unreliable which provides a confidence measure without requiring technical knowledge.

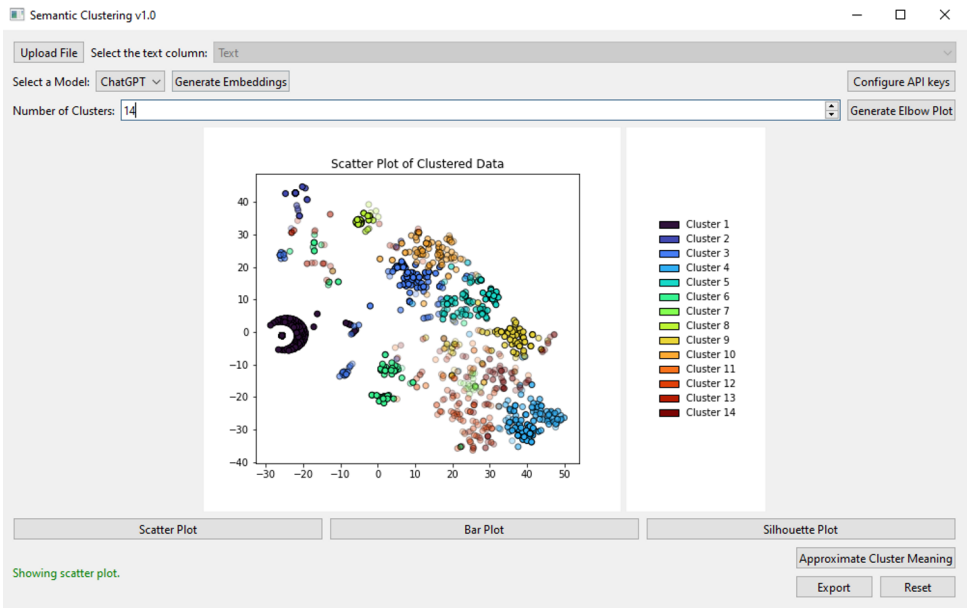


Figure 1: Example of the clustering interface with grouped survey data.

Results

The toolbox was tested using real-world survey data provided by the IFSAR research institute.

The evaluation showed:

- Major themes were successfully identified
- Results were interpretable and useful for exploratory analysis
- The interface was well received by non-technical users

Limitations

- The quality of clustering depends on input text clarity (e.g. spelling or off-topic responses).
- Without labeled data, results must be interpreted subjectively.
- Performance and visualization clarity decrease with very large datasets or high cluster counts

Future Potential

The toolbox provides a strong foundation for further improvements:

- Integration of more preprocessing features (e.g. spell checking)
- Extended export options and integration with external systems
- Support for larger datasets through performance optimization

1 Introduction

This chapter introduces the motivation and context of the research. It outlines the problem the thesis aims to solve, formulates the central research questions, and defines the objectives of the work. Additionally, the chapter provides an overview of the structure of the thesis and the methodological approach used, with a focus on the Design Science Research paradigm.

1.1 Motivation and Problem Statement

Researchers at IFSAR regularly conduct surveys that include open-ended questions. These questions are a valuable part of the data collection process, as they allow participants to express their opinions freely and potentially raise issues that the researchers may not have previously considered. Unlike multiple-choice questions, which restrict responses to a predefined set of options, open-ended questions can uncover new insights and provide more qualitative data. However, this flexibility comes at a cost. The analysis of open-ended responses is typically manual, subjective, and highly time-consuming. While a multiple-choice question may have only a handful of distinct responses, open-ended questions can generate a vast number of unique answers — often as many as there are participants. This significantly complicates the task of aggregating and interpreting the data, especially when dealing with large sample sizes.

1.2 Objective of the Thesis

The objective of this thesis is to develop and evaluate a software tool that supports the analysis of open-ended survey responses through semantic clustering.

Semantic clustering refers to the grouping of textual data based on the underlying meaning of the responses rather than surface-level similarity such as keyword overlap Aggarwal and Zhai [2012]. Modern natural language processing (NLP) techniques — such as sentence embeddings — make it possible to represent individual responses as high-dimensional vectors that capture semantic content. By applying clustering algorithms to these representations, thematically similar responses can be grouped together, even if they use different words or phrasing.

This approach is particularly suited to the problem at hand: researchers at IFSAR must manually process a wide variety of open-ended survey answers, which is both time-consuming and difficult to scale. Semantic clustering offers a way to automate part of this process by surfacing patterns and grouping related answers, thus enabling faster, more structured analysis without sacrificing the richness of qualitative data.
The tool is developed using the Design Science Research (DSR) methodology, with the goal of creating a practically relevant and scientifically grounded artifact. The scope of this work is limited to short- to medium-length German open-ended survey responses and focuses on clustering as an aid for human interpretation, not as a replacement for expert analysis.

1.3 Research Questions

To guide the development and evaluation of the proposed artifact, this thesis investigates the following research questions:

> **RQ1: How can a semantic clustering toolbox be developed to support non-technical users in analyzing open-ended survey responses?**
> **RQ2: How stable are the resulting clusters across multiple initializations, and how can this stability be visualized to improve user interpretation?**

These questions focus on assessing the capability of modern semantic techniques to improve the analysis of qualitative survey data by automatically identifying meaningful clusters of responses.

Structure of the Thesis

The remainder of this thesis is structured as follows:

- **Chapter 2** presents the theoretical and technical foundations, including an overview of semantic clustering, natural language processing techniques, and relevant related work.
- **Chapter 3** describes the research methodology based on the Design Science Research framework and outlines the evaluation approach.
- **Chapter 4** details the conceptual design of the semantic clustering tool, explaining architectural decisions and algorithm choices.
- **Chapter 5** focuses on the implementation of the artifact, highlighting key components and development challenges.
- **Chapter 6** reports the evaluation results, analyzing the effectiveness and practical relevance of the tool.
- **Chapter 7** provides a discussion of the findings, contributions, limitations and offers recommendations for future work.
- **Chapter 8** concludes the thesis and summarizes its main outcomes.

2 Theoretical and Technical Foundations

This chapter provides the necessary background for the development of the semantic clustering toolbox. It begins with an overview of DSR as applied in computer science, outlining its relevance to this project. Following this concepts and recent advances in semantic clustering and automated data analysis are presented to lay the theoretical groundwork for the artifact's design. The chapter then reviews prior research focused on semantic clustering of survey responses, highlighting the gaps and opportunities addressed by this thesis. Finally, the chapter introduces the technologies employed in the implementation of the semantic clustering toolbox.

2.1 Design Science Research in Computer Science

Design Science Research is an approach that focuses on creating and evaluating artifacts to solve practical problems Hevner et al. [2004]. It is especially suited for computer science research where designing and building useful software tools or systems is the main goal. Although this thesis does not follow a formal DSR process from start to finish, it is inspired by the core principles of DSR:
Iterative problem-centered research, artifact creation and evaluation.
The project involved identifying a real-world challenge — the manual analysis of open-ended survey responses —, designing and implementing a semantic clustering tool to address this challenge, and evaluating the tool's effectiveness in practice.
By adopting this approach, the research ensures that the developed solution is both technically sound and applicable to the needs of researchers analyzing survey data.

2.2 Fundamentals and State of the Art in Semantic Clustering

Semantic clustering is the process of grouping textual data based on the meaning or semantics of the content, rather than relying solely on surface-level features such as keyword overlap. This allows for the identification of thematically similar texts even when different vocabulary or phrasing is used.

The practice of clustering texts by semantic meaning is well-established, with roots tracing back to early research in computational linguistics. For instance, Karen Spärck Jones Spärck Jones [1965] pioneered methods that used manually constructed thesauri to infer semantic similarity. Modern approaches have significantly advanced this field by leveraging high-dimensional vector representations generated by neural language models.

Vector Representations of Text At the core of semantic clustering lies the representation of text as numerical vectors that encode semantic information. With the advent of distributed representations, approaches like Word2Vec Mikolov et al. [2013] and GloVe Pennington et al. [2014] made it possible to encode words into dense vector spaces based on their co-occurrence statistics in large corpora.
However, word embeddings alone are insufficient for tasks that require understanding whole sentences or documents. Sentence-level embedding models such as Sentence-BERT (SBERT) Reimers and Gurevych [2019] extend the capabilities of models like BERT Devlin et al. [2019] to produce semantically meaningful representations of longer texts, making them suitable for clustering applications.

Clustering Algorithms Once texts are represented as vectors, clustering algorithms can be applied to detect groupings of semantically similar responses. The K-means algorithm, used in this thesis, is one of the most popular methods due to its computational efficiency and intuitive geometric interpretation. It partitions data into k clusters by minimizing the within-cluster variance. More advanced techniques, such as hierarchical clustering and density-based methods like DBSCAN, are also commonly applied, but K-means remains a strong baseline for many unsupervised NLP tasks.

2.3 Prior Work on Semantic Clustering of Survey Responses

One of the most relevant contributions in recent years comes from Esmaeilzadeh et al. [2022], who propose an end-to-end framework for extracting insights from open-ended survey responses. Their approach is centered around the use of sentence embeddings generated by pre-trained transformer models, which are then clustered using context-aware clustering techniques. The results are visualized through word clouds and representative terms, making the output more interpretable for users.

While this work demonstrates that semantic clustering can effectively support the interpretation of qualitative survey data, it exhibits several limitations that are relevant to the present thesis. First, the framework is designed for English-language responses and does not consider multilingual data. Second, the implementation is briefly described as an on-device mobile application, but no source code or interface details are provided, making it difficult to assess its accessibility for non-technical users. Third, the framework does not address the stability or consistency of generated clusters which is an important aspect when drawing conclusions from qualitative data groupings.

Other tools and frameworks, such as ClustVis Metsalu and Vilo [2015], ClustrLab2k13 Patil [2023], and Hugging Face's text-clustering repository von Werra and Allal [2024], offer components of semantic clustering workflows, including visualization and embedding-based clustering. However, these tools typically serve general-purpose scenarios and are not specialized for working with qualitative survey data. They often require command-line interaction or technical expertise, assume that the input is already embedded in vector form, lack integrated sentiment analysis and do not provide built-in assessments of cluster reliability through a stability metric, which limits their practical usability for users without a technical background.

Rather than adopting one of these tools directly, this thesis builds upon their core ideas and presents a simplified pipeline for semantic clustering that targets short to medium-length German survey responses. The toolbox prioritizes accessibility, offering a graphical interface that enables non-technical users to explore the clustering results. In addition to the core clustering functionality, it includes visualizations for evaluating cluster stability, thereby contributing to the interpretability of the findings.

Challenges in Semantic Clustering of Survey Responses Open-ended survey responses are typically short, unpredictable, and highly variable in language and style. Multilingual or dialectal variations, such as Swiss German, add further complexity to semantic representation. Additionally, determining the appropriate number of clusters is not trivial and often requires iterative approaches. Finally, clusters must be interpretable

and useful for human analysts, balancing automation with expert review.
This thesis addresses these challenges by combining state-of-the-art language models with a clustering approach tailored to the specific needs of qualitative survey analysis, focusing on usability and practical relevance.

2.4 Technological Background

This section provides an overview of the important technologies that serve as the foundation of the toolbox developed in this thesis. It covers essential concepts from NLP, methods for representing text as embeddings, techniques to measure similarity between these representations, and algorithms used for clustering and dimensionality reduction. Together these components form the computational foundation required to semantically group and analyze open-ended survey responses.

2.4.1 Natural Language Processing

Natural Language Processing is a subfield of artificial intelligence and focuses on how computers can process and analyze human language. In the context of this thesis, NLP techniques are used to transform textual survey responses into representations that machines can analyze. Processing natural language poses challenges due to the ambiguity and complexity of the human languages, especially when dealing with informal or mixed dialect texts such as Swiss German.

2.4.2 Text Representation Using Embeddings

Understanding and comparing textual data requires representing text in a form that computer models can understand and process effectively. Modern approaches use **embeddings**, which are dense numerical vectors that capture the semantic meaning of text elements such as words, sentences, or entire documents. Unlike traditional methodes based on word counts, embeddings position semantically similar texts closer together in a high-dimensional vector space, enabling models to recognize meaning beyond exact word matches.

Recent advances in NLP have been driven by transformer-based models like BERT Devlin et al. [2019] and its sentence-focused variant SBERT Reimers and Gurevych [2019]. These models generate context-aware embeddings that take into account the surrounding words and the overall sentence structure, producing semantic representations well-suited for tasks like classification or clustering. By leveraging these high dimensional embeddings, semantic clustering algorithms can group text responses based on meaning, even when they use different wording or phrasing. Embedding vectors typically range from a few hundred to over a thousand dimensions. Higher dimensions can differentiate more subtle semantic features, but increase computational cost.

2.4.3 Similarity Measures

To cluster texts based on their embeddings, it is necessary to quantify the similarity between vectors. Cosine similarity is one of the most commonly used metrics in NLP, as it measures the cosine of the angle between two vectors, capturing semantic similarity regardless of vector magnitude.

In this work, the clustering algorithm relies on Euclidean distance, as used by the standard **K-means** implementation in scikit-learn [2025a]. While Euclidean distance is sensitive to vector magnitude, the embeddings used in this system are L2-normalized, this means each vector is scaled to have a unit length of 1. In such cases, squared Euclidean distance becomes mathematically proportional to cosine dissimilarity Patel [2020]. This makes Euclidean-based clustering behaviorally equivalent to cosine-based clustering for normalized embeddings, allowing for efficient and semantically meaningful groupings.

2.4.4 Clustering Algorithms

Clustering is the task of grouping similar data points, in this case sentence embeddings, into tight groups called **clusters** based on a similarity or distance metric. In the context of this thesis, the objective is to group survey responses with similar meanings, even if their surface forms vary significantly.

This thesis uses the **K-means** algorithm which is a widely adopted and computationally efficient clustering method. K-means aims to partition the n data points $\{x_0, x_1, \ldots, x_n\}$ into k clusters $C = \{C_1, C_2, \ldots, C_k\}$ by minimizing the within-cluster variance, specifically the sum of squared Euclidean distances between each data point x_i and its assigned cluster centroid μ_j (see Equation (1)). The algorithm is initialized with k random centroids and iteratively updates cluster assignments and centroids until convergence scikit-learn [2025b].

$$\sum_{i=0}^{n} \min_{\mu_j \in C} (\|x_i - \mu_j\|^2) \tag{1}$$

Adapted from scikit-learn [2025b]

Although K-means requires the number of clusters k to be specified in advance, it performs well on normalized text embeddings, making it a suitable choice for this system. Its non-deterministic initialization behavior also enables repeated trials, which are valuable for evaluating cluster stability which is a central aspect of this thesis.

2.4.5 Dimensionality Reduction

High-dimensional data such as sentence embeddings can be challenging to interpret for humans. Dimensionality reduction techniques transform this complex data into a lower-dimensional space, making it easier to analyze and visualize while preserving meaningful structure. Common methods include Principal Component Analysis (PCA) Jolliffe [2002], which identifies directions of greatest variance, and t-distributed Stochastic Neighbor Embedding (t-SNE) Van der Maaten and Hinton [2008], which focuses on maintaining local similarities between data points.
In this thesis, dimensionality reduction is applied to visualize clustering results.

3 Research Methodology and DSR Approach

This chapter outlines the methodology for this project, focusing on the DSR framework. It details how DSR principles and processes were applied to iterativly develop, evaluate, and refine the semantic clustering toolbox. The chapter begins by introducing the DSR cycle and explaining why it was selected.

3.1 Design Science Research Cycle

Design Science Research is a problem-solving methodology that emphasizes the creation and evaluation of artifacts intended to solve identified problems in a relevant context. According to Hevner Hevner [2007], DSR can be understood through a three-cycle model, consisting of:

- **Relevance Cycle**: Connects the research to the application environment and stakeholders, ensuring the problem and solution are grounded in real-world needs.
- **Rigor Cycle**: Grounds the research in existing theories, frameworks, and knowledge bases to provide a solid scientific foundation.
- **Design Cycle**: Encompasses the iterative process of building and evaluating the artifact to address the problem and implement user feedback.

These three cycles interact continuously throughout the research process, ensuring a balance between practical relevance, theoretical rigor, and effective design.

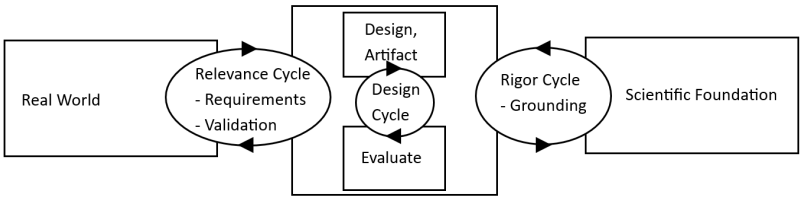


Figure 2: The Three Cycle View of DSR (adapted from Hevner [2007]).

Design Science Research was chosen as the projects methodology because it aligns closely with the objective of this thesis: to develop a practical and user-centric artifact that addresses a real-world problem. DSR emphasizes the creation, evaluation, and iterative refinement of an artifact in collaboration with real users, making it suitable for research that aims to generate a usable tool rather than purely theoretical insights.

3.2 Requirements Analysis and Objective Definition

As part of the relevance cycle in the DSR process, a requirements analysis was conducted to ensure that the developed artifact effectively addresses real user needs. This analysis involved an interview with a researcher from IFSAR to define the problem domain and artifact requirements, complemented by a review of the project assignment.

The primary objective of this research is to design and develop a user-centric toolbox for semantic text clustering that addresses the needs of non-technical users. This artifact aims to facilitate the analysis of open-ended survey responses through interpretable clustering results. The core objectives of the project are:

- To design an intuitive and accessible clustering toolbox tailored to practical requirements.
- To provide comprehensive documentation that supports user adoption and effective utilization.
- To investigate the impact of random initialization on the stability of k-means clustering results
- To develop visualization techniques for conveying this stability.

3.2.1 Functional Requirements

To develop an artifact that fits these objectives, a set of functional and non-functional requirements was established. These requirements were gathered from the interview with the IFSAR researcher[1] and the provided project assignment Tödtli [2025]. Table 1 summarizes the main functional requirements that define the core capabilities of the semantic clustering toolbox.

ID	Name	Description
FR1	Upload file	Enable users to upload Excel files containing survey responses.
FR2	Generate embeddings	Transform uploaded texts into numerical embedding vectors using a pre-trained model.
FR3	Cluster embeddings	Apply the K-means clustering algorithm to group similar embeddings.
FR4	Export results	Allow exporting of clustering results to an Excel file.
FR5	Visualize clusters	Provide visualizations such as scatter plots to interpret cluster distributions.
FR6	Sentiment analysis	Estimate sentiment for each cluster using a language model.
FR7	Cluster stability	Visualize stability of clustering across multiple runs with different initializations.

Table 1: Functional Requirements

[1] A paraphrased summary of the interview is provided in Appendix A

3.2.2 Non-Functional Requirements

In addition to the functional capabilities, certain non-functional requirements guide the design and implementation of the toolbox. These address usability and technology constraints, ensuring the artifact meets user standards. Table 2 outlines these requirements.

ID	Name	Description
NFR1	User Accessibility	The toolbox should be easy to use, even for non-technical users.
NFR2	Technology Stack	The toolbox must be implemented in Python.

Table 2: Non-Functional Requirements

3.2.3 Project Scope

The requirements are categorized into a Minimum Viable Product (MVP) and planned additions to guide development priorities. This structuring allows for iterative improvements based on user feedback and evolving project needs according to the design cycle.

Minimum Viable Product The MVP defines the essential functionality to deliver a working and usable system:

- FR1 – Upload Excel files containing the survey responses.
- FR2 – Generate embeddings from texts using a suitable embedding model.
- FR3 – Cluster embeddings with the K-means algorithm.
- FR4 – Export clustering results to Excel.

Planned Additions These goals represent planned additions to improve usability and analytical capabilities:

- FR5 – Visualization of clusters for exploratory analysis.
- FR6 – Sentiment estimation per cluster using language models.
- FR7 – Analysis and visualization of cluster stability over multiple runs.
- Allow selection between different embedding service providers.
- Enable saving of API keys for embedding services across sessions.
- Implement a recommendation system to suggest an optimal number of clusters (e.g. via elbow plot).

3.3 Evaluation Methods

Evaluating the effectiveness of the semantic clustering toolbox involves a combination of clustering quality assessment and user validation checks, consistent with the principles of DSR.

Clustering Quality Assessment Internal clustering metrics, such as the silhouette score Rousseeuw [1987] and cluster stability are used to assess cluster quality. While these metrics do not provide an absolute measure of correctness in a unsupervised setting, they offer indicators of clustering quality.

User Validation User feedback is essential to validate the interpretability of the clustering results and the usability of the toolbox interface. Interviews and demo sessions with the researcher from IFSAR, help ensure the clusters visualizations are understandable and that the interface is intuitivly supporting the intended user workflow.

Requirement Validation A requirement validation using a real-world dataset of open-ended survey responses, provided by IFSAR, demonstrates the applicability and usefulness of the toolbox. This involves running the full user workflow — from data upload through clustering and visualization — and assessing whether the arifact fulfills the functional requirements.

Together, these evaluation methods provide an understanding of the artifact's performance. Clustering quality assessment aligns with the **rigor cycle** by grounding evaluation in established metrics. User validation supports the **relevance cycle** by ensuring practical utility and user expectations are met. Finally, the requirement validation reflects the **design cycle** by enabling real-world testing and iterative refinement of the artifact.

4 Design of the Artifact

This chapter presents the conceptual design of the semantic clustering toolbox, grounded in the iterative and user-centric principles of DSR. The design aims to support non-technical users in interpreting open-ended survey responses by providing an accessible and intuitive interface.

A simplified domain model was created during requirements analysis to support the definition of the user workflow and clarify the main entities and interactions in the system (Figure 3).

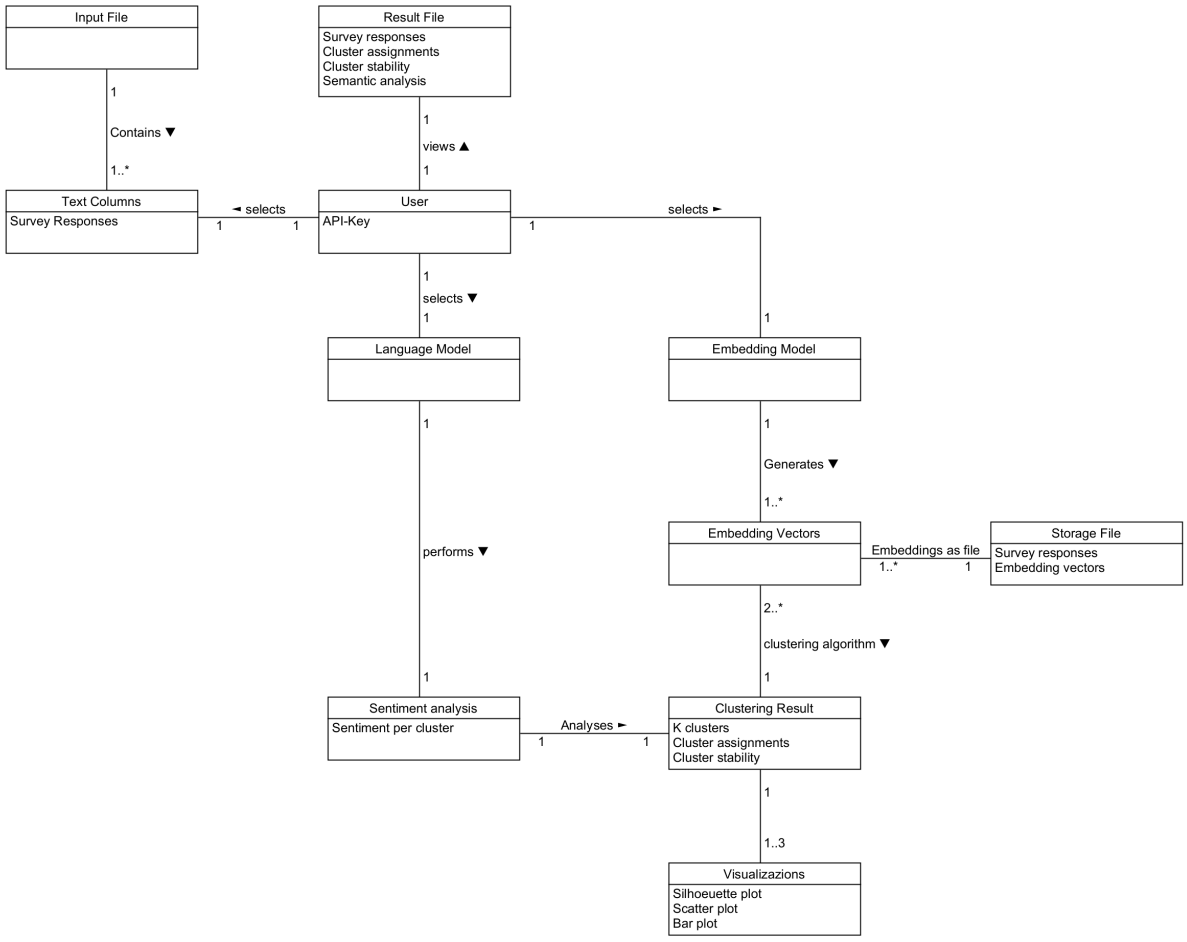


Figure 3: Domain model of the semantic clustering process.

Based on this model, a user workflow was defined that serves as the foundation for the artifact's design cycle. It comprises the following steps:

1. **Loading Input File:** Uploading open-ended survey responses stored in Excel files.
2. **Column and Model Configuration:** Choosing the relevant column of responses and selecting embedding and language model providers.
3. **Embedding:** Generating embeddings from text data using embedding models accessed through external APIs.
4. **Clustering:** Clustering the embeddings to group semantically similar responses.
5. **Visualization:** Analyzing cluster stability and interpreting cluster quality.
6. **Sentiment Analysis:** Interpreting cluster sentiment via language models accessed through external APIs.
7. **Export:** Exporting results for further analysis.

The following sections elaborate on the system architecture, user interface, accessibility features and the design rationale for each stage of the workflow.

4.1 System Architecture

The semantic clustering toolbox is designed as a mostly self contained application. Its architecture separates user interaction, processing logic, and external services. This modular design supports extendability and maintainability. A high-level architectural overview is shown in Figure 4.

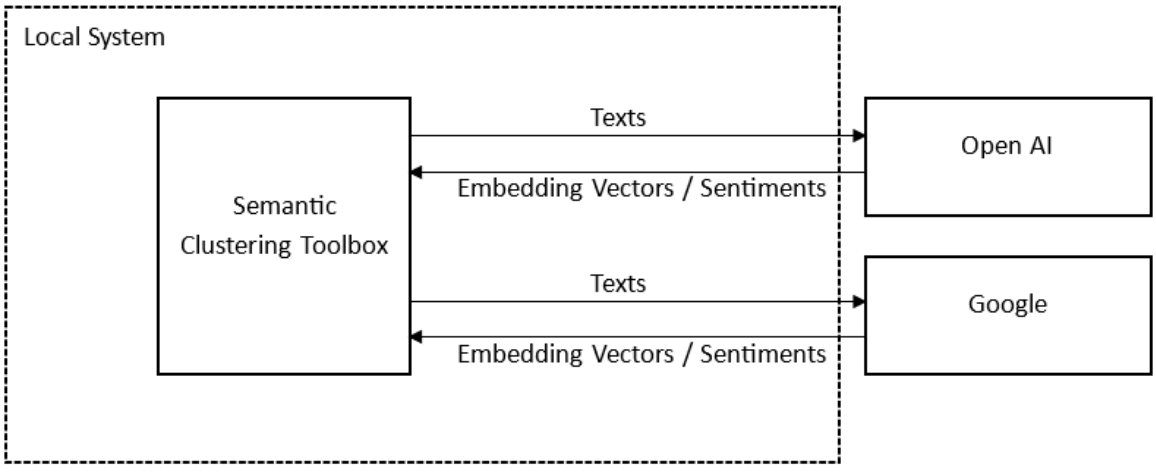


Figure 4: High-level architecture of the semantic clustering toolbox.

At a conceptual level the system architecture consists of three elements:

- **User Interface:** Provides a graphical interface for non-technical users to configure inputs, monitor progress, and review results.

- **Processing Logic:** Performs the semantic clustering workflow by using various Python libraries and external services.
- **External APIs:** Accesses external language and embedding model APIs through an abstracted layer. This design separates provider-specific logic behind a common interface which enables support for multiple model providers and simplies future expansion.

To prevent the graphical user interface (GUI) from freezing during compute-intensive operations or API calls, long-running tasks are executed asynchronously.

4.2 User Interface

In compliance with NFR1 the GUI is designed to support non-technical users throughout the entire semantic clustering workflow. Built using the PySide6 The Qt Company [2025] framework it offers a clear flow guiding users from data loading to final export without requiring any command-line interaction. PySide6 was selected for its extensive set of user-friendly widgets and cross-platform compatibility making it ideal for designing an artifact for non-technical users.

User feedback is delivered through progress messages and error reports for issues such as incompatible file types or failed API calls. This design ensures a guided and intuitive experience that helps users successfully navigate the semantic clustering process.

4.3 Accessibility

To improve usability for non-technical users, the toolbox is designed with installers and additional documentation (see Appendix E).
Two installation options are provided to simplify setup:

- Platform-specific installation scripts that automate environment setup and dependency installation, assuming Python 3.11 is pre-installed.
- A standalone Windows executable that enables a simple double-click-to-run experience without requiring Python or manual dependency management.

In addition, a user guide is included to provide step-by-step instructions from installation through operation. The guide features annotated screenshots to assist users who may be unfamiliar with technical details.

4.4 Input Handling

The toolbox accepts both **.xlsx** files, as outlined by FR1, and **.parquet** files, which the toolbox uses to store previously generated embeddings. A graphical file dialog is provided to allow users to select a file from their local system. Once loaded, the file is parsed using the **pandas** library pandas [2025]. Its used due to its widespread use in data science applications and support for Excel input. The toolbox then prompts the user to select one column containing the open-ended responses to be analyzed.

4.5 Model Configuration

Following successful file and column selection the toolbox prompts users to configure the embedding and language models used throughout the semantic clustering process. The model configuration is presented through a dropdown menu. Users are able to choose between supported providers (OpenAI and Google) for both embedding and cluster sentiment analysis tasks. Providers and models are selected by the criteria discussed in Section 4.6 and Section 4.9. API key management is handled via a dedicated dialog accessible from the interface.

During development, an additional input field was considered to allow users to switch between model variants offered by the supported providers. This feature would have enabled users to choose between different models based on dataset size or desired embedding quality. However, to avoid overwhelming non-technical users unfamiliar with the distinctions between models, it was decided not to implement this option. This design choice prioritizes a ease of use over advanced customization.

4.6 Embedding Generation

Once the text column is selected and model configuration is complete, the toolbox prompts the user to begin the embedding phase, in which each response is converted into a numerical vector representation using the selected model providers external API. For a brief explanation of embeddings, refer to Section 2.4.2.

The resulting vectors are stored in memory within a **pandas** DataFrame. This temporary representation is automatically cleared if the user changes the input column, uploads a new file, or initiates a new embedding process. To support reuse and avoid redundant API calls users are prompted to save the generated embeddings to a local **.parquet** file. This format was chosen for its fast serialization performance.

Before executing embedding requests, the toolbox presents a cost estimation dialog that displays the estimated token usage and expected API costs based on the current dataset and selected model. This allows users to stay informed about their spending and prevents accidental high-cost operations.

Since embedding involves potentially long-running API requests, this is one of the processes that are executed asynchronously. Users are notified of the progress through a progress bar while error handling ensures that API-related issues such as connectivity failures are caught and reported within the interface.

4.7 Clustering Algorithm

In line with functional requirement FR3, the toolbox clusters the embedded survey responses using the **K-means** algorithm. The algorithm itself was previously described in Section 2.4.4.

Users initiate clustering by specifying the desired number of clusters k and selecting any of the visualization options. To support the selection of k the toolbox includes an elbow plot that visualizes how the within-cluster variance changes across a range of k

values, helping identify the point where increasing k yields minimal improvement. Additionally the toolbox integrates an extended variant of K-means known as **X-means** Pelleg et al. [2000], which automatically estimates a suitable value for k by optimizing the Bayesian Information Criterion. This suggested cluster count is presented to users as guidance for those less familiar with tuning clustering parameters for a complex dataset.

4.8 Cluster Visualization

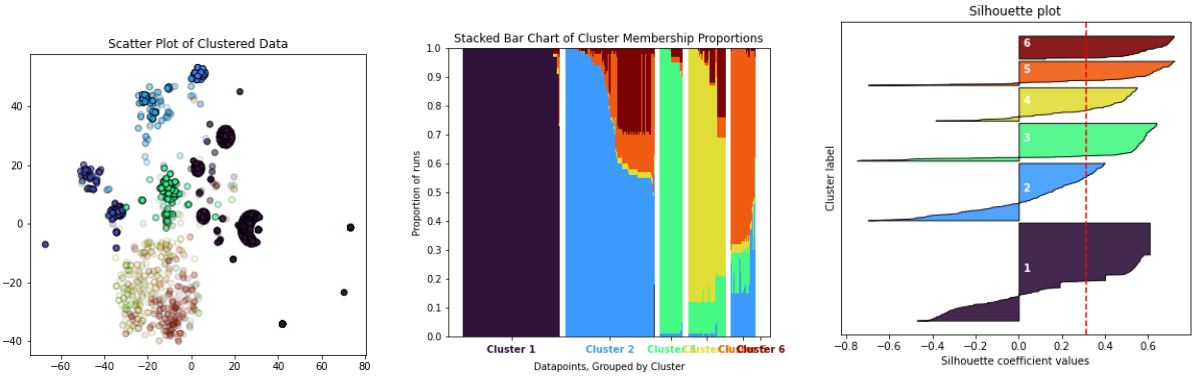
To support interpretability of the clustering results and fulfill functional requirements FR5 and FR7, the toolbox provides multiple visualization options that allow users to explore both the quality and stability of clustering outcomes.
Three types of plots are integrated into the design:

Scatter Plot : This plot presents the clustered embeddings projected into two dimensions using a dimensionality reduction technique called **t-SNE** Van der Maaten and Hinton [2008]. By preserving local relationships from the high-dimensional space t-SNE allows K-means clusters to appear as distinct groups in a two dimensional illustration. Transparency is used to encode the stability of cluster assignments across multiple runs which helps users identify data points with uncertain or inconsistent clustering.

Bar Plot : The bar chart represents each data point's membership proportions across clusters over multiple clustering runs. Data points are grouped by their most frequent cluster assignment which illustrates the overall stability of clusters and potential overlaps between semantically similar groups.

Silhouette Plot : Illustrates the silhouette coefficient for each data point, reflecting how similar it is to its own cluster compared to the nearest neighboring cluster. This plot allows users to assess the tightness and separation of clusters Rousseeuw [1987].

These visualizations render the clustering results interpretable to human users and support the evaluation of both cluster quality and stability



(a) Scatter Plot **(b)** Bar Plot **(c)** Silhouette Plot

Figure 5: Illustrative examples of the visualizations included in the toolbox.

These visualizations are shown here in simplified form to illustrate the intended design. Full-size versions and detailed explanations are provided in Section 5.5

4.9 Sentiment Interpretation

To provide additional interpretability of the clustered survey responses, the toolbox includes an optional **sentiment analysis** step. Rather than evaluating sentiment at the level of individual responses, the system estimates sentiment on a cluster level based on the content of the cluster. This design decision further supports the goal of forming semantically meaningful groupings with summaries as defined by FR6.

Each cluster is interpreted using an external **language model** that infers the overall sentiment expressed across the clusters responses. The resulting sentiment is displayed alongside each cluster in the export, allowing users to quickly identify a clusters approximate sentiment.

As mentioned by the IFSAR researcher in the MVP demonstration interview (see appendix B), this functionality significantly increases the toolboxes usability and marks another step in the relevance cycle.

4.10 Result Export

To support analysis and reporting the toolbox includes functionality for exporting the clustering results as specified in FR4. Users can export the final results to a structured Excel file containing the following information per data point:

- The original survey response.
- The assigned cluster label.
- The confidence of the assignment across multiple runs.
- The sentiment associated with the cluster (if sentiment analysis is enabled).

These outputs provide a human-readable summary of the clustering results that can be integrated into existing analysis workflows. This export capability was deemed as an important feature by the ISFAR researcher during requirement gathering and contributes to the utility and usability of the designed artifact.

4.11 Model Selection Rationale

The toolbox currently supports using models from OpenAI and Google. They were selected based on embedding quality, pricing, and API availability.

Embedding Models

This section outlines the main embedding models considered and explains the rationale for their inclusion.

OpenAI: OpenAI's embedding models have been shown to work effectively with clustering methods such as K-means Petukhova et al. [2025]. In January 2024, OpenAI released the *text-embedding-3* series OpenAI [2024] which significantly outperforms the *ada-002*

model used by Petukhova et al. [2025]. This series includes two variants: *text-embedding-3-small* and *text-embedding-3-large*. According to the MTEB benchmark Enevoldsen et al. [2025], the large model offers slightly better performance but incurs a roughly sixfold increase in token cost. Given that survey responses are typically short and embedding quality is critical for semantic clustering, *text-embedding-3-large* was chosen as the default model.

Google: At the time of this research, Google's *text-embedding-004* model Lee et al. [2024] was ranked at the top of the MTEB leaderboard. In March 2025, it was superseded by the experimental *gemini-embedding-exp-03-07* (also referred to as *gemini-embedding-001*) Lee et al. [2025]. Despite its strong performance the new model remains in a experimental phase and is subject to restrictive rate limits Google [2024], making it unsuitable until the full release. The toolbox therefore defaults to *text-embedding-004* with a transition to *gemini-embedding-001* recommended once it exits the experimental stage. Notably *text-embedding-004* is scheduled for deprecation in November 2025 Google [2025c].

Language Models

For the sentiment analysis descibed in 4.9 and by FR6, the toolbox utilizes large language models (LLMs) from the same providers as the embedding models to simplify user experience by avoiding multiple API keys.

OpenAI: As of April 2025, OpenAI's *GPT-4.1* series represents the latest generation of models. The toolbox defaults to the *gpt-4.1-mini* variant which balances high-quality language understanding with significantly improved response speed and lower costs compared to the full *gpt-4.1* model OpenAI [2025a]. This choice aligns with the design goal of delivering timely insights while maintaining cost efficiency for practical deployment.

Google: Google's *Gemini 2.5* series is a recent advancement in language models Google [2025a], with the *gemini-2.5-flash* model currently in limited preview since May 2025 Google [2025b]. Due to its experimental status and usage restrictions the toolbox presently utilizes the stable *gemini-2.0-flash* model. This choice ensures reliability during the research phase with an recommended update to the latest model upon full release.

This model selection approach supports the overarching DSR principles of relevance and rigor by balancing state-of-the-art capabilities with practical constraints such as cost.

5 Implementation

This chapter details the practical realization of the semantic clustering toolbox introduced in the design chapter. Building on the architectural and functional specifications previously outlined, the concrete technologies used to implement the toolbox are now described. The implementation follows the three-layered structure defined during design:

- **User Interface:** A graphical interface built to support non-technical users in configuring inputs, initiating processing, and reviewing clustering results.
- **Processing Logic:** The core backend workflow responsible for transforming raw survey data into meaningful clusters, leveraging NLP libraries and clustering algorithms.
- **External APIs:** An abstraction layer that integrates language and embedding models from third-party providers, ensuring modularity and ease of future extension.

The following sections present the technology stack, describe the implementation of each core component, and discuss deployment considerations.

5.1 Technology Stack

The semantic clustering toolbox is implemented in Python, fulfilling the non-functional requirement NFR2 that specifies Python as the development language.
The primary technologies and libraries used in the implementation include:

- **Python 3.11:** The release of Python used during the development.
- **PySide6:** A Python binding of the Qt toolkit used to build the GUI.
- **Pandas:** Used for data import preprocessing and manipulation of survey responses.
- **NumPy:** Provides support for numerical operations and array handling during data processing and algorithm implementation.
- **Scikit-learn:** Supplies the used implementation of the K-means clustering algorithms, along with evaluation metrics like silhouette scores.
- **Matplotlib:** Used for data visualization like plotting cluster distributions that are displayed within the GUI.
- **OpenAI and Google Generative AI APIs:** Accessed via respective Python client libraries to obtain language model embeddings and other NLP functionalities.

Additional tools such as tiktoken for token management, dotenv for environment configuration, and PyInstaller for packaging are discussed in their respective sections.

Together these technologies fulfill the toolbox's requirements for data processing, user-friendly interaction, and integration with external services.

5.2 Implementation of Core Components

This section presents the detailed implementation of the semantic clustering toolbox's core components, following the architectural structure defined in Section 4.1 of the design chapter.
The implementation is divided into three main parts:

- **User Interface**, which facilitates interaction for non-technical users.
- **Processing Logic**, which manages data transformation and clustering.
- **External API Integration**, which handles communication with language and embedding models.

Each part is discussed in detail below, describing the implementation of the design.

5.2.1 User Interface

The GUI of the semantic clustering toolbox is designed to provide a simple and intuitive experience for non-technical users. It is implemented using the **PySide6** framework as outlined in Section 4.2

The interface includes file selection dialogs, parameter input fields, progress indicators, and a result visualization panel. These components guide users through the workflow described in Chapter 4 without requiring programming knowledge.

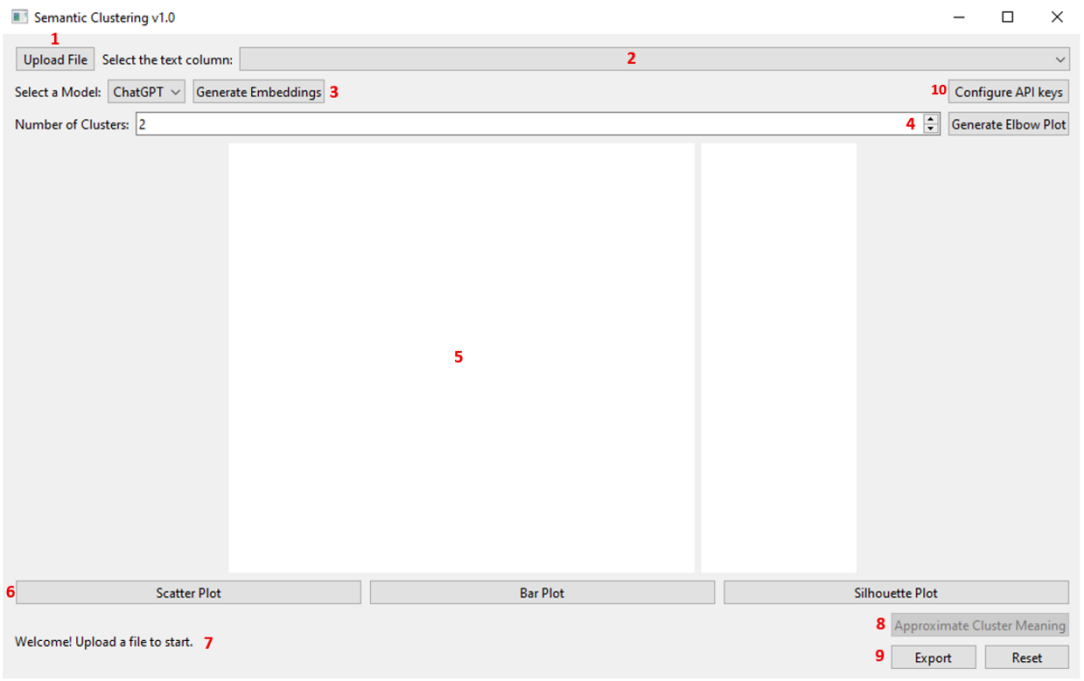


Figure 6: Screenshot of the semantic clustering toolbox graphical user interface.

Figure 6 illustrates the main window of the toolbox, showcasing the important elements:

1. **File Selection Dialog:** Allows users to browse and select input survey files in the supported formats.

2. **Column Selector Dropdown:** Enables the user to choose the specific column containing open-ended survey responses for analysis.
3. **Model Provider Selection:** Allows users to choose the model provider (discussed in Section 4.5).
4. **Clustering Parameters Panel:** Users can configure the number of clusters k and display the elbow plot, as described in Section 4.7.
5. **Results Visualization Area:** Displays the different supported plots.
6. **Visualization Creation Buttons:** Generates the respective visualizations described in Section 4.8.
7. **Status Message and Progress Bar:** Displays status messages to guide the user and provides real-time feedback during asynchronous operations.
8. **Sentiment Analysis Button:** Starts the sentiment analysis process.
9. **Export Button:** Enables saving the clustering results to external files.
10. **API Key Dialog:** Opens the API key dialog.

API Key Dialog

This dialog, presented in Figure 7, provides a dropdown to select the provider, a input field for entering the corresponding API key, and buttons to add or remove stored keys. API keys are saved locally as environment files within an **API_keys** directory which enables persistent authentication across sessions. To avoid accidental user errors the interface validates key content (e.g. ASCII-only), confirms overwrites and displays error messages for any issues encountered during saving or deletion.

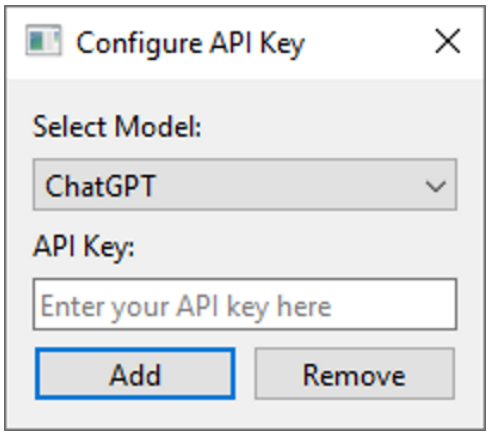


Figure 7: Screenshot of the API Key Dialog interface.

5.2.2 Processing Logic

This subsection details the core internal workflow of the semantic clustering toolbox, covering the steps from raw data preprocessing to clustering and sentiment analysis.

The internal architecture supporting this workflow is illustrated in the UML class diagram shown in Figure 8, which depicts the main classes such as the main application class and worker classes that handle asynchronous tasks.

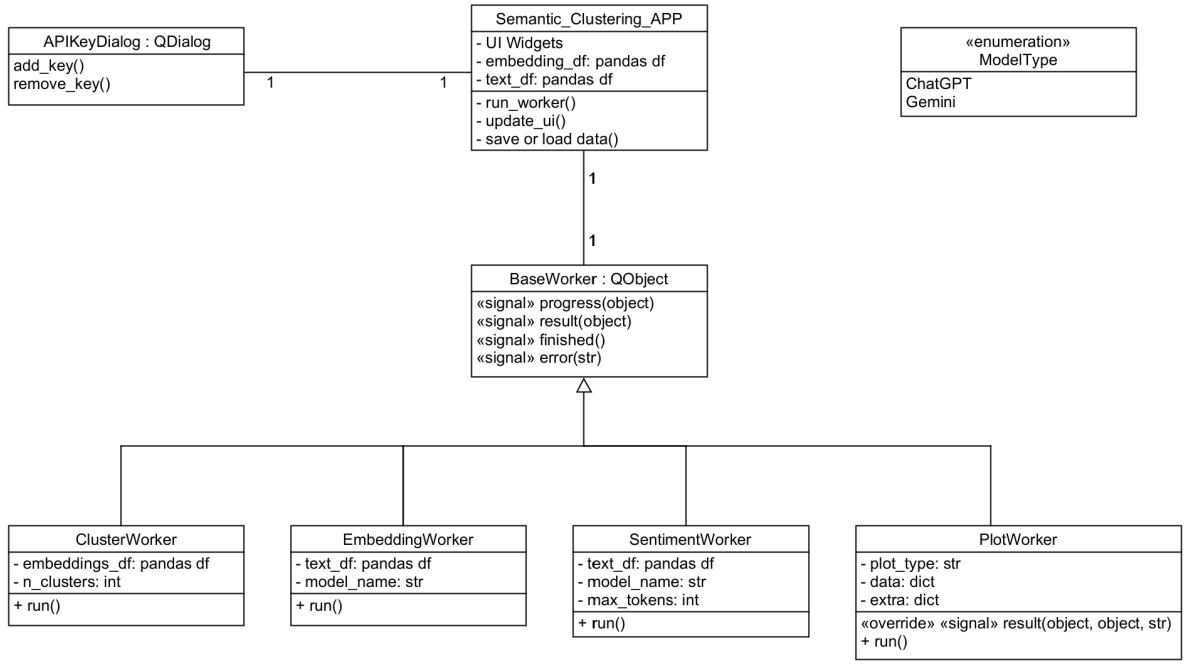


Figure 8: UML Class Diagram

The subsequent sections describe these components in context as the processing pipeline progresses.

Asynchronous Execution

To prevent the GUI from freezing during long-running operations, the toolbox executes these tasks asynchronously using worker classes built on PySide6's *QThread*. All workers, except the **ClusterWorker**, report progress to the GUI which enables the use of a progress bar to inform users about the task completion status.

Four specialized worker classes handle computationally intensive or API-related tasks:

- **EmbeddingWorker** — Handles embedding API calls.
- **ClusterWorker** — Performs the clustering process.
- **SentimentWorker** — Executes semantic analysis API calls.
- **PlotWorker** — Generates the requested visualizations.

The main application class, **Semantic_Clustering_APP**, temporarily disables the GUI functions during asynchronous operations to prevent conflicting interactions.

Preprocessing

To prepare this data for semantic processing the following preprocessing steps are applied:

- Rows with missing values or empty strings in the selected column are removed.
- All remaining entries are cast to string format.

To prevent errors due to unclusterable input the system verifies that the selected column contains more than one unique value. If this condition is not met the user is prompted to select a different column.

Finally column names are normalized by replacing special characters and truncating the name to a maximum of 40 characters. This ensures consistent and safe names for output files and intermediate results.

Embedding Generation

The preprocessed textual responses are converted into vector embeddings using external language models accessed through the OpenAI and Google Generative AI APIs. The embedding generation process is executed asynchronously by the **EmbeddingWorker** class (see Figure 8), this follows the procedure outlined in Section 4.6.

To handle large datasets efficiently and reduce API usage costs, the system implements a batching mechanism that splits the input texts into batches of 100 entries. These batches are formatted as lists to comply with the API requirements.

Clustering

Once embeddings are generated, the toolbox applies the K-means clustering algorithm to group semantically similar responses, fulfilling functional requirement FR3. The clustering process is executed asynchronously by the **ClusterWorker** class. The number of clusters k can be set by the user or estimated automatically using the X-means method described in Section 4.7.

To assess the stability of the clustering results the algorithm is executed 100 times on the same dataset with different random initializations. For the purpose of this chapter, each of these executions is referred to as a single clustering run. This approach supports the evaluation of cluster stability and independence from initialization, as outlined in FR7. Since cluster stability analysis is one of the main objective of this thesis, the specific algorithm used for this evaluation is described separately in Section 5.3.

The toolbox utilizes the scikit-learn [2025a] implementation of k-means with the following parameters to ensure truly random centroid initializations:

- **n_clusters** – Specified by the toolbox user.
- **init** – Set to "**random**" for fully random initializations.
- **n_init** – Set to **1** to ensure only a single initialization per clustering run.

All other parameters use their default values.

Sentiment Analysis

As described in FR6 the toolbox performs sentiment analysis on survey responses using language models accessible through external APIs. This process is managed asynchronously by the **SentimentWorker** class, and was previously outlined in Section 4.9.

The **SentimentWorker** aggregates texts within each cluster into a newline-separated string, adding entries iteratively until reaching a maximum token limit (default: 3000 tokens). The semantic analysis leverages a zero-shot learning approach, where the model is given only instructions on the task without any labeled examples. This approach avoids introducing bias from predefined example data. The exact prompt template used for this analysis is detailed in Section C of the appendix.

Error Handling

Error handling mechanisms are implemented throughout the processing pipeline. Invalid inputs, API failures, and unexpected conditions trigger error messages that are displayed within the GUI.

5.2.3 External API Abstraction

The toolbox interfaces with two external API providers: OpenAI and Google Generative AI. It uses the official OpenAI [2025b] Python library to access OpenAI's *embedding* and *chat completion* endpoints, and the official Google APIs [2025] GenAI Python library to interact with Google Gemini's *embed_content* and *generate_content* endpoints. These APIs are used in two stages of the processing pipeline:

1. generating vector embeddings from raw input texts.
2. producing semantic summaries or sentiment labels for clustered data.

Further details on these processes are provided in Section 5.2.2.

Custom Model Providers

Throughout the toolbox, models are referenced exclusively via the **ModelType** enum. This abstraction allows for the integration of new language or embedding models through a simple two-step process:

1. Inclusion of the new model identifier within the **ModelType** enum.
2. Implementation of the corresponding embedding or semantic analysis function, following the specifications or templates provided in the *model_api_clients.py* file.

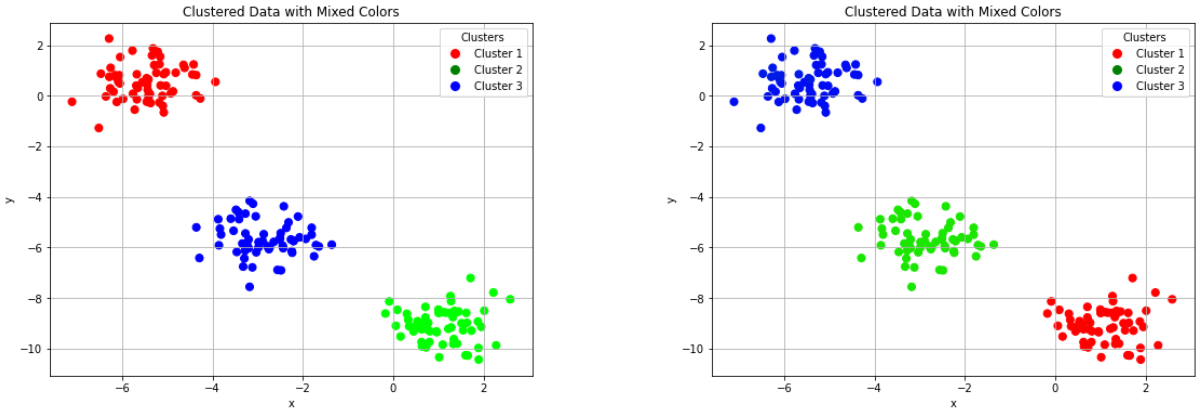
Each custom function is linked to the appropriate model using decorators which automatically register them in two internal dictionaries: one for embedding functions and another for semantic analysis functions. This implementation decouples the core logic of the toolbox from model-specific details, ensures it does not need to be altered when adding custom models.

5.3 Cluster Stability Analysis

An essential goal of this project, as outlined in FR7, is to make clustering stability both measurable and visually interpretable by evaluating how consistent the K-means results are across multiple runs with different initializations. The project assignment by Tödtli [2025] defined it as:
"The goal of this research is to make the concepts of "stability" and the "independence of clusters from the clustering initialization" tangible and to develop a quantitative measure that captures this stability".

5.3.1 Cluster Matching Challenge

A considerable challenge in evaluating clustering stability is identifying corresponding clusters across different clustering runs. Figure 9 illustrates this issue using a simplified example: two sets of clustered datapoints, each produced by a separate clustering run. Although the cluster labels differ between clustering A and clustering B, an observer can easily see that the red cluster in clustering A corresponds to the blue cluster in clustering B. In practice datasets are rarely this clear-cut. Real-world data often contains overlapping or ambiguous cluster boundaries, and individual datapoints may be assigned to different clusters across runs. Under such conditions identifying stable, corresponding clusters becomes substantially more complex.



Clustering A *Clustering B*

Figure 9: Comparing two clusterings

5.3.2 Cluster matching Algorithm

To address this issue, a sequence-based algorithm is used. This algorithm builds on the assumption that corresponding clusters do exist across runs, and therefore does not perform well when the underlying data does not support a meaningful clustering structure, for example in cases where the datapoints form a network in which each point has approximately equal distance to all its neighbors.
The core idea is that if stable, corresponding clusters exist, then there must be datapoints near the center of each cluster that consistently remain grouped together across clustering runs. The greater the number of such consistently co-assigned datapoints, the more stable the cluster is considered to be.

In the context of this section, a **sequence** refers to the ordered list of cluster assignments that a single datapoint receives across multiple clustering runs. For example, if a datapoint is assigned to cluster 2 in the first run, cluster 3 in the second, and cluster 2 again in the third, its sequence would be [2, 3, 2]. The algorithm collects these sequences from all datapoints and counts how frequently each unique sequence occurs. The most common sequences indicate groups of datapoints that consistently cluster together across runs, revealing stable and corresponding clusters.

The algorithm proceeds by reassigning cluster labels based on the k most frequent assignment sequences, where k is the total number of clusters. It assigns the label *0* to the cluster corresponding to the most frequent sequence, label *1* to the second most frequent, and so on. Next, it iterates through these frequent sequences in order. For each sequence, if a label at any run index r is **greater than** the assigned label for that sequence, the algorithm performs a simultaneous replacement of all occurrences of the larger label with the assigned label in both the cluster results and the sequence matrix, for that run. If the label at run r is smaller than or equal to the assigned label, no action is needed. This means the label was either already correctly reassigned in an earlier step or is already correct. A pseudo-code implementation of the relabeling algorithm is depicted in Algorithm 1.

Notation:

- N: number of datapoints
- R: number of clustering runs
- k: number of clusters
- $C \in \mathbb{N}_0^{N \times R}$: cluster assignments for each datapoint across R runs
- $S \in \mathbb{N}_0^{k \times R}$: k most frequent cluster assignment sequences across R runs
- i: new cluster label $(0 \leq i < k)$
- r: run index (column index of C and S), $(0 \leq r < R)$
- s: sequence index (row index of S), $(0 \leq s < k)$

Algorithm 1 Cluster Matching via Frequently Assigned Sequences

Require: $C \in \mathbb{N}^{N \times R}$ cluster results ▷ N datapoints clustered over R runs
Require: $S \in \mathbb{N}^{K \times R}$ frequency ordered sequences ▷ k most frequent sequences across R
Ensure: Labels in C are relabeled for consistency across different run

```
i = 0
for each s in S do
    for each run r in s do
        if i < S_sr then
            Simultaneously replace i → S_sr and S_sr → i in row r of C
            Simultaneously replace i → S_sr and S_sr → i in row r of S
        end if
    end for
    i += 1
end for
return updated C
```

Result: Cluster labels in C are relabeled so that corresponding clusters are assigned the same label across all runs.
Section F in the Appendix presents a small example illustrating the algorithm.

5.3.3 Cluster Assignment

Each datapoint is assigned to the cluster it was most frequently associated with across all clustering runs. This approach can result in some clusters not being assigned any datapoints, as no datapoint had that cluster as its most frequent assignment. As a result, the final output may contain fewer than the original k clusters, although all k clusters will still appear in the percentage breakdown.

5.3.4 Cluster Stability

Cluster stability measures the consistency of cluster assignments across multiple runs of the clustering algorithm. It provides insight into how reliably datapoints belong to the same cluster, helping to identify stable, well-defined groups versus ambiguous or overlapping ones. By quantifying stability, users can better interpret the robustness of the clustering results and prioritize clusters or datapoints for further analysis.

$$\text{stability} = M \times \left(1 - \alpha \times \frac{d-1}{k-1}\right)$$

Here, M represents the **main cluster percentage**, which is the proportion of clustering runs in which a datapoint was assigned to its most frequent cluster. d denotes the number of distinct clusters the datapoint was assigned to, k is the total number of clusters, and α is a weighting parameter between 0 and 1 that controls the penalty for being assigned to multiple different clusters. By default the toolbox uses $\alpha = 0.2$ to avoid over-penalizing such cases.

The stability of each cluster is then quantified as the average stability of its assigned datapoints, reflecting how consistently those datapoints were grouped together across runs.

5.4 Packaging and Installation

To facilitate user setup, two installation methods were implemented:

- **Platform-specific installation scripts** automate the environment setup and dependency installation. These scripts assume Python 3.11 is pre-installed and handle all necessary configuration steps.
- **Standalone Windows executable**, created using *PyInstaller* Cortesi [2025] to provide a simple double-click-to-run experience. This executable bundles the application along with all required dependencies, eliminating the need for users to install Python or manually manage dependencies.

The source code and installation scripts are available on GitLab at `https://gitlab.ost.ch/lukas.derungs/ba`

5.5 Visualization

As previously described in Section 4.8 the toolbox generates three different plot types for the evaluation of clustering results. These were implemented using Python's *matplotlib* and *scikit-learn* libraries.

Scatter Plot

Clustered data points are projected into two dimensions using the *scikit-learn* implementation of t-SNE. The plot colors points by their most frequent cluster assignment and uses transparency to encode stability: Points consistently assigned to the same cluster across multiple runs are more opaque, while less stable points are more transparent. This visualization helps identify cluster structure and unstable or ambiguous datapoints.
Figure 10 shows a scatter plot generated by the toolbox.
Clusters 1 through 6 exhibit relative stability which is indicated by many opaque points, while clusters 7 through 14 are all mixed with mostly transparent datapoints indicating instability and possibly simmilar topics.

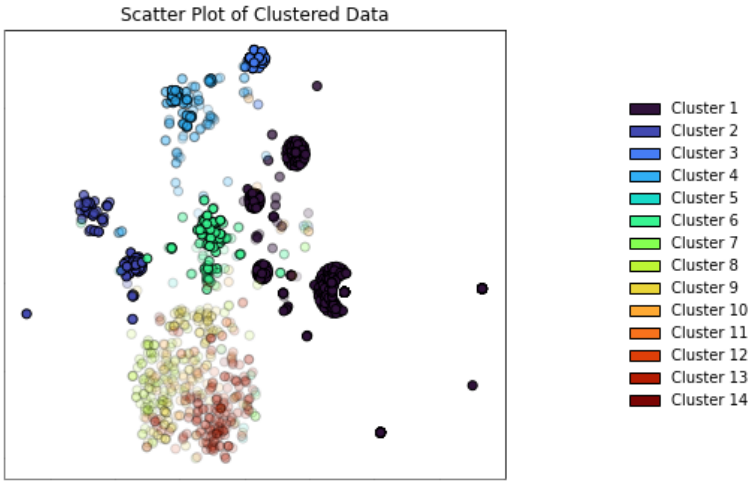


Figure 10: Example of a scatter plot visualizing cluster assignments.

Stacked Bar Plot

This visualization shows each datapoint's cluster membership proportions across multiple clustering runs using stacked bars. The Y-axis indicates the percentage of runs in which a datapoint was assigned to each cluster. Along the X-axis, datapoints are grouped by their most frequent cluster assignment and sorted within each group based on how consistently they belonged to that cluster. This plot displays the stability of cluster assignments and reveals overlapping clusters.

Figure 11 shows a bar plot generated by the toolbox.
Cluster 1 appears stable, with all its datapoints assigned to it in almost 100% of runs.
Clusters 2 and 3 show a stable core, but also include datapoints that were assigned to other clusters frequently.
Cluster 4 shows substantial overlap with Cluster 2, as its datapoints were assigned to Cluster 2 in approximately 40%
Cluster 5 seems to be mostly stable with overlaps with clusters 2 and 4.
Cluster 6 appears very unstable, all of the datapoints assigned to it only belonged to cluster 6 in about 35% to 40% of runs.

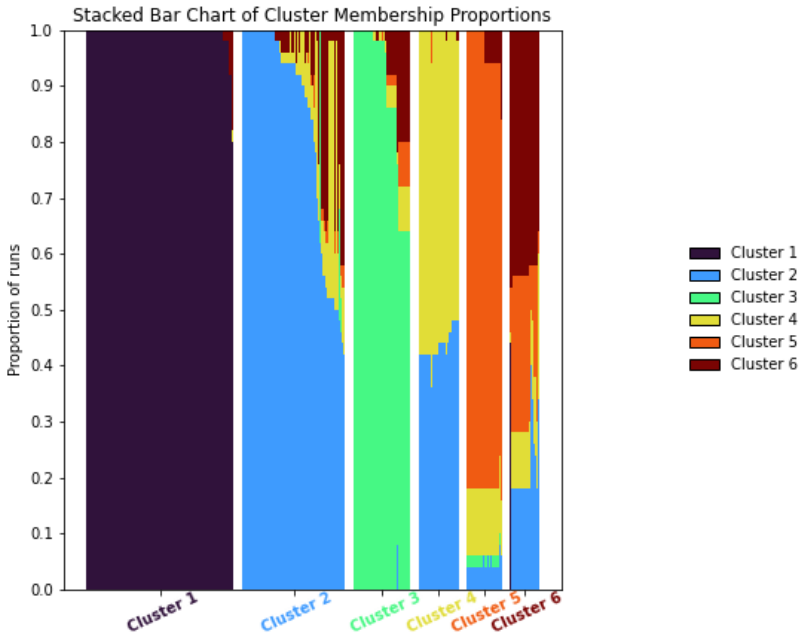


Figure 11: Cluster stability plot.

Silhouette Plot

The silhouette coefficient is computed for each datapoint to assess cluster cohesion and separation. The silhouette scores are calculated using *scikit-learn*'s *silhouette_samples* and *silhouette_score* functions. Bars representing scores are grouped and colored by cluster, with the average silhouette score indicated by a red vertical line. This plot visually conveys cluster quality.

Figure 12 shows a silhouette plot generated by the toolbox.
In this example all clusters contain a few data points with negative silhouette scores which indicates that these points are closer to the centroids of other clusters than to their own.

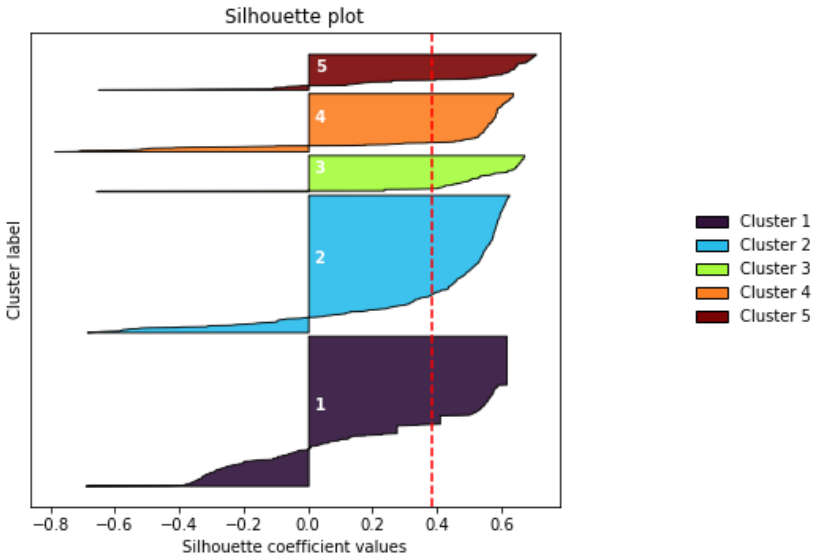


Figure 12: Example of a silhouette plot visualizing cluster quality.

All plots use consistent color schemes and labeling enabling intuitive comparison across the different visualizations.

6 Evaluation

This chapter evaluates the semantic clustering toolbox developed in this thesis. Following the DSR methodology outlined in Section 3.3, the artifact is assessed through both technical metrics and user-centered validation. The evaluation includes a requirement validation based on authentic survey data and aims to measure the artifact's real-world usability.

6.1 Evaluation Setup

The evaluation was designed to assess the toolbox's technical performance and usability. This follows the DSR framework by incorporating both the rigor and relevance cycles.

Evaluation Objectives

The main objectives were:

- To assess the clustering quality and interpretability of the generated results.
- To validate the usability and workflow of the toolbox for non-technical users.

Dataset

A real-world dataset of open-ended survey responses provided by IFSAR was used for testing. The dataset consists of 1600 anonymized responses related to various topics and reflects a realistic usage scenario.

Methods

The evaluation employed the following methods:

- **Clustering quality:** Evaluated using internal metrics such as silhouette score and cluster stability.
- **User validation:** Conducted through an interview and demo session focusing on usability and interpretability.
- **Requirement validation:** A full end-to-end workflow was tested using the provided survey data to simulate real-world use.

6.2 Clustering Quality Assessment

To evaluate how effectively the toolbox identifies meaningful groupings in textual data, two internal clustering metrics were used: silhouette score Rousseeuw [1987] and cluster stability. While these metrics do not establish correctness in an unsupervised setting, they serve as established indicators of clustering quality and help determine whether the artifact produces interpretable, semantically coherent clusters. This assessment aligns with the rigor cycle of DSR by grounding evaluation in quantitative methods.

The silhouette scores ranged between 0.1 and 0.4 depending on the number of clusters,

with higher values generally observed for lower k. These scores suggest low to moderate tightness and separation among clusters. Cluster stability was assessed by running the K-means algorithm 100 times with different random initializations, as described in Section 5.2.2. The stability analysis revealed that a few larger clusters exhibited high stability (average stability above 0.75) which indicating consistent structure. However most clusters showed low average stability (many below 0.2) which suggests that their boundaries are less robust.

These results suggest that frequently mentioned topics tend to form larger, more stable clusters which makes them easily recognizable across runs. In contrast less commonly expressed sentiments are distributed among smaller, less stable clusters. This indicates that the toolbox is especially effective at highlighting dominant themes while still offering a degree of structure for analyzing more nuanced or minority perspectives.

6.3 User Validation

User validation was conducted through two semi-structured interviews with a researcher from IFSAR, one during the requirements gathering phase (see Section 3.2) and another after the MVP was implemented. These interactions were essential for aligning the artifact with real user needs, evaluating its practical usability and ensuring the artifact fulfills NFR1.

Feedback After MVP Demonstration

In a follow-up meeting the MVP was demonstrated to the same researcher [2]. The user confirmed that the core functionality (including file upload, clustering and export features) met expectations and addressed the primary use case of analyzing open-ended survey responses.
Some areas for improvements were also identified:

- **Sentiment analysis:** The researcher expressed strong interest in leveraging a language model to provide sentiment information for each cluster. The researcher noted that this feature would substantially increase the artifacts usefulness and help with future analysis of the exported results.
- **Column selection:** Manual selection of the relevant column for clustering was requested as most of their surveys are stored in multi-column Excel files.

This feedback provided valuable input for refining the artifact in subsequent iterations and confirmed that the MVP fulfilled the primary workflow needs of its intended users.

[2] A paraphrased summary of the interview is provided in Appendix B

6.4 Requirement Validation

The requirement validation outlined in Section 3.3 tested the semantic clustering toolbox against the functional requirements defined in Section 3.2. The following summarizes how each requirement was met using the IFSAR survey dataset.

- **FR1 - Upload files:** The toolbox successfully imported the 1600 survey responses from an Excel file.
- **FR2 - Generate embeddings:** Text responses were transformed into embeddings using the chosen model.
- **FR3 - Cluster embeddings:** K-means clustering was applied successfully.
- **FR4 - Export results:** Clustering outcomes were exported to an Excel file.
- **FR5 - Visualize clusters:** Visualizations were able to be generated.
- **FR6 - Sentiment analysis:** The integrated language model provided sentiment estimations for clusters.
- **FR7 - Cluster stability:** Multiple runs confirmed cluster stability.

This evaluation confirmed the artifact's compliance with the core functional requirements, demonstrating its practical readiness for real-world use. An interpretation of the clustering results of the requirement validation can be found in Section 7.1

7 Discussion and Reflection

This chapter provides a discussion and reflection on the findings from the evaluation of the semantic clustering toolbox and the development process. Building on the results and user feedback presented in the previous chapter, it interprets the significance of these outcomes and their implications for the project's objectives.

Furthermore, it addresses the limitations encountered during the study and outlines opportunities for future improvements. Through this reflective analysis, the chapter aims to provide an understanding of the project's contributions, challenges, and potential future directions.

7.1 Interpretation of Evaluation Results

The evaluation results indicate that the semantic clustering toolbox fulfills its intended purpose of supporting the analysis of open-ended survey responses. While the silhouette scores suggest only moderate cluster quality, the system proved effective in identifying dominant themes within the dataset. This can be seen in the results of the case-study. The generated clusters were considered largely meaningful and often shared a consistent sentiment. However there were also responses that did not fit the topic of the cluster that they were assigned. This typically occurred when the response in question got assigned to two clusters the same amount of times or when it contained spelling errors or off-topic statements that didnt seek to anwser the survey question. Further it is important to acknowledge that, in the absence of ground truth labels, the correctness of these groupings remains subjective and open to interpretation.

As no additional feedback from IFSAR on the final toolbox version was received by the time of writing, it is not possible to conclusively evaluate whether the user interface satisfies NFR1. However, given that only minor GUI changes were made since the MVP demonstration, and that the interface was previously well-received, it can be reasonably assumed that the system remains intuitive and usable in a real-world setting. Moreover, non-IFSAR users have tested the toolbox and were able to use it effectively with the provided documentation (see Appendix E), providing informal validation of its usability.

Taken together, these findings suggest that the toolbox is well-suited for identifying overarching patterns in survey responses while still offering a degree of structure for less common or nuanced sentiments. This aligns with the artifact's intended role of supporting exploratory analysis, even if the precision of individual assignments is not always perfect.

7.2 Limitations

While the evaluation demonstrated the toolbox's practical utility, several limitations became apparent during development and testing.

First, the clustering performance depends heavily on the quality of the input data. Responses that are off-topic, written in dialect, or contain spelling errors can reduce the effectiveness of the embedding process and through that clustering accuracy. This was

evident in the requirement validation, where some datapoints were misclassified due to off-topic input.

Second, the silhouette score provided only a limited view of clustering quality when large values for k were used. Even when the silhouette score suggested a poor clustering, the exported result often appeared to form meaningful groupings upon manual inspection. Without access to ground truth labels, it remains difficult to objectively assess whether the clusters reflect meaningful groupings. This introduces a level of subjectivity that cannot be fully avoided in unsupervised data analysis.

Third, the language model often failed to follow instructions during sentiment inference. It had a tendency to list topics instead of focusing solely on sentiment, even when explicitly prompted not to do so. This behavior might be mitigated through the use of a specially fine-tuned model or by applying more prompt engineering.

Another limitation relates to the handling of user interviews. While insights from the IFSAR researcher were recorded in the form of paraphrased notes, no formal transcripts were produced. A full transcription could provide a more traceable basis for evaluating usability and impact.

These limitations show areas where the accuracy of the artifact could be further improved.

7.3 Future Work

While the current version of the semantic clustering toolbox achieves its core goals, several areas for future improvements have been noted during development and evaluation.

- **Handling of input:** The clustering quality is affected by responses that include spelling mistakes, dialect, or irrelevant content. Future versions could incorporate more thorough preprocessing, such as spell-checking or filtering of low-quality responses to improve embedding quality.
- **Evaluation with labeled data:** Due to the lack of ground truth the evaluation remains largely subjective. Having a dataset that was correctly clustered by a group of experts beforehand would enable an objective validation of clustering performance.
- **Performance optimization:** For larger numbers the plotting of visualziation and clustering can be time intensive which could be reduced with more efficient implementations.
- **Improved visualization:** While the current visualizations effectively convey the intended insights for smaller numbers of clusters k, larger values (greater than 20) can lead to cluttered graphics. Future improvements could include interactable visualizations or alternative methods that avoid displaying every single cluster to maintain clarity.
- **Advanced features:** Many possible features were omitted in favour of ease of use. Future versions could include an "advanced" mode that offers greater control for experienced users (see appendix G for details).

- **Export capabilities:** Additional export options for the generated visualizations could be implemented to support use cases where users wish to save visual results.
- **Update Models:** As mentioned in Section 4.11, the landscape of Embedding and Language models evolves quickly, a new iteration should definitely include newer, updated models.

These directions could offer promising future steps for extending the flexibility, performance and usability of the toolbox in future iterations.

8 Summary

This thesis presented the design, development and evaluation of a semantic clustering toolbox aimed at supporting non-technical users in analyzing open-ended survey responses. Motivated by the challenges of expediting a time-consuming data analysis task in research settings, the project followed a Design Science Research methodology to iteratively build and assess a functional artifact. The toolbox integrates state-of-the-art language and embedding models, the K-means clustering algorithm and dimensionality reduction techniques into a user-friendly application featuring simple visualizations. Its core functionalities — including file upload, semantic embedding, clustering, sentiment analysis, visualization and export — were validated through a requirement validation using real survey data and user feedback. Evaluation results indicate that while subjective interpretation remains a factor, the system successfully identifies major themes in survey data and enables meaningful exploratory analysis. In addition, the evaluation of cluster stability across multiple initializations showed that larger clusters representing dominant topics tend to remain consistent, supporting confidence in the interpretability of the results. Several limitations were identified, such as sensitivity to input quality and the lack of objective evaluation due to missing ground truth labels. These findings informed a set of directions for future work, including improved preprocessing, more robust evaluation methods, and support for advanced features.

Overall, the semantic clustering toolbox makes a practical contribution to the field of semantic clustering of textual data by offering an accessible and extensible tool for exploring textual survey data, particularly for non-technical users.

References

Charu Aggarwal and Chengxiang Zhai. A survey of text clustering algorithms. *Mining Text Data*, 08 2012. doi: 10.1007/978-1-4614-3223-4_4.

David Cortesi. Pyinstaller documentation, 2025. URL `https://pyinstaller.org/en/stable/`. Accessed: 2025-06-01.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL `https://arxiv.org/abs/1810.04805`.

Kenneth Enevoldsen, Isaac Chung, and et. al Imene Kerboua. Mmteb: Massive multilingual text embedding benchmark. *arXiv preprint arXiv:2502.13595*, 2025. doi: 10.48550/arXiv.2502.13595. URL `https://arxiv.org/abs/2502.13595`.

Saeid Esmaeilzadeh, Marek Rei, and Sebastian G. Bošnjak. Providing automatic insights into open-ended responses with semantic clustering. In *Artificial Intelligence in Education: 23rd International Conference, AIED 2022, Durham, UK, July 27–31, 2022, Proceedings, Part I*, pages 526–532. Springer International Publishing, 2022. ISBN 9783031116445. doi: 10.1007/978-3-031-11644-5. URL `http://dx.doi.org/10.1007/978-3-031-11644-5`.

Google. Google embedding models, 2024. URL `https://ai.google.dev/gemini-api/docs/models#text-embedding`. Accessed: 2025-06-01.

Google. Google gemini 2.5 announcement blog, 2025a. URL `https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/#gemini-2-5-thinking`. Accessed: 2025-06-06.

Google. Google gemini 2.5 flash documentation, 2025b. URL `https://cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-5-flash`. Accessed: 2025-06-06.

Google. Google models retirement plan, 2025c. URL `https://cloud.google.com/vertex-ai/generative-ai/docs/learn/model-versions`. Accessed: 2025-06-01.

Google APIs. Google apis github organization. `https://github.com/googleapis/python-genai`, 2025. GitHub repository, accessed 2025-06-03.

Alan Hevner. A three cycle view of design science research. *Scandinavian Journal of Information Systems*, 19, 01 2007.

Alan R Hevner, Salvatore T March, and et. al Park. Design science in information systems research. *MIS quarterly*, 28(1):75–105, 2004.

Ian T Jolliffe. *Principal component analysis for special types of data*. Springer, 2002.

Jinhyuk Lee, Zhuyun Dai, and et al. Xiaoqi Ren. Gecko: Versatile text embeddings distilled from large language models, 2024. URL `https://arxiv.org/abs/2403.20327`.

Jinhyuk Lee, Feiyang Chen, and et al. Sahil Dua. Gemini embedding: Generalizable embeddings from gemini, 2025. URL `https://arxiv.org/abs/2503.07891`.

Tauno Metsalu and Jaak Vilo. Clustvis: a web tool for visualizing clustering of multivariate data using principal component analysis and heatmap. *Nucleic Acids Research*, 43(W1):W566–W570, 05 2015. ISSN 0305-1048. doi: 10.1093/nar/gkv468. URL https://doi.org/10.1093/nar/gkv468.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013. URL https://arxiv.org/abs/1301.3781.

OpenAI. Openai release, 2024. URL https://openai.com/index/new-embedding-models-and-api-updates/. Accessed: 2025-06-01.

OpenAI. Openai platform - 4.1 index. https://openai.com/index/gpt-4-1/, 2025a. Accessed: 2025-06-02.

OpenAI. openai-python: Openai python api library. https://github.com/openai/openai-python, 2025b. GitHub repository, accessed 2025-06-03.

pandas. pandas library, 2025. URL https://pandas.pydata.org/. Accessed: 2025-06-05.

Ajay Patel. Relationship between cosine similarity and euclidean distance, 2020. URL https://ajayp.app/posts/2020/05/relationship-between-cosine-similarity-and-euclidean-distance/. Accessed: 2025-06-05.

Ashish Gupta Patil. Clustrlab2k13. https://github.com/code2k13/ClustrLab2k13, 2023. GitHub repository, Accessed: 2025-05-28.

Dan Pelleg, Andrew Moore, et al. X-means: Extending k-means with e cient estimation of the number of clusters. In *ICML'00*, pages 727–734. Citeseer, 2000.

Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://aclanthology.org/D14-1162/.

Alina Petukhova, João P. Matos-Carvalho, and Nuno Fachada. Text clustering with large language model embeddings. *International Journal of Cognitive Computing in Engineering*, 6:100–108, 2025. ISSN 2666-3074. doi: https://doi.org/10.1016/j.ijcce.2024.11.004. URL https://www.sciencedirect.com/science/article/pii/S2666307424000482.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019. URL https://arxiv.org/abs/1908.10084.

Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

scikit-learn. scikit kmeans reference, 2025a. URL https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html. Accessed: 2025-06-02.

scikit-learn. scikit documentation, 2025b. URL https://scikit-learn.org/stable/modules/clustering.html#k-means. Accessed: 2025-06-02.

Karen Spärck Jones. Experiments in semantic classification. *Mechanical Translation and Computational Linguistics*, 8(1):1–10, 1965. URL https://mt-archive.net/MT-1965-Sparck-Jones.pdf.

The Qt Company. Pyside6 documentation, 2025. URL https://doc.qt.io/qtforpython/. Accessed: 2025-06-01.

Beat Tödtli. Project assignment, 2025. see Apendix E, provided by OST.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Leandro von Werra and Lobna Ben Allal. text-clustering. https://github.com/huggingface/text-clustering, 2024. Accessed: 2025-05-28.

Appendices

A Paraphrased Interview Summary

This appendix provides a paraphrased summary of an interview conducted with a researcher from IFSAR during the early project phase. The interview aimed to clarify domain-specific challenges, user needs, and functional expectations for the semantic clustering toolbox. While no formal transcript was recorded, the following points summarize the requirements discussed:

- **Problem Context:** The researcher pointed out the recurring task of analyzing open-ended survey responses.
- **User Base:** The tool should be usable by researchers with limited programming experience.
- **Core Features:** The ability to upload survey data, generate semantic clusters, and export results were considered core functionalities.
- **Visualization:** The inclusion of visualizations was considered important for assessing the quality of clustering results.
- **Data Format:** Survey responses were typically stored in Excel files, establishing this as the preferred input format.

This paraphrased summary serves as the basis for the functional and non-functional requirements presented in Section 3.2.

B Paraphrased Feedback After MVP Demonstration

A follow-up meeting was held to demonstrate the MVP to the same IFSAR researcher. The following points summarize the user's reactions and suggestions:

- The user confirmed that the core functionality met expectations and expressed confidence in its practical utility.
- There was strong interest in the integration of a language model for the analysis of cluster sentiment.
- The ability to manually select the relevant column for clustering was identified as a necessary improvement.
- It was noted that some survey responses contained Swiss-German dialect, suggesting a potential challenge for language models.

C Semantic Analysis Prompt

This section shows the prompt used to guide the language model in summarizing clusters by extracting their shared meaning in a concise sentence, avoiding listing individual details.
The prompt:

You will be shown a list of semantically similar text items. Your task is to distill the shared meaning into a short sentence (ideally under 15 words) that captures the core intent or unifying concept. Focus on why these items belong together, not what they each individually discuss. Avoid listing examples or summarizing each item. Instead, express the underlying motivation, value, or problem they collectively address.
cluster item A
cluster item B
cluster item C
cluster item D
cluster item E
What is the shared topic or meaning?

D Tools and Technologies Used

This table provides an overview of the main tools, libraries and technologies employed throughout the project.

Tool	**Purpose of Use**
Visual Studio Code	Writing and managing the LaTeX document
Spyder 5.0.4 (Python IDE)	Developing and executing scripts
ChatGPT	Assistance with document structure, phrasing and spelling
Python 3.11.1	Main programming language for implementation
PySide6	Python library used for developing the GUI
Pandas, Scikit-learn, Matplotlib	Python libraries for data processing, clustering, dimensionality reduction and plotting
OpenAI / Google Generative APIs	Generating semantic text embeddings and inferring sentiment
Git	Version control during development

Table 3: Tools and technologies used during the project

E Additional Documentation

The userguide, the installation instructions and the project assignment are provided on the OST SharePoint: SharePoint

F Example of Cluster Label Algorithm

This section depics a small example of the custer label algorithm described in Section 5.3.2.

Algorithm 2 Cluster label algorithm for reference

Require: $C \in \mathbb{N}^{N \times R}$ cluster results ▷ N datapoints clustered over R runs
Require: $S \in \mathbb{N}^{K \times R}$ frequency ordered sequences ▷ k most frequent sequences across R
Ensure: Labels in C are relabeled for consistency across different run

```
i = 0
for each s in S do
    for each run r in s do
        if i < S_sr then
            Simultaneously replace i → S_sr and S_sr → i in row r of C
            Simultaneously replace i → S_sr and S_sr → i in row r of S
        end if
    end for
    i += 1
end for
return updated C
```

Let the initial cluster result matrix C and the top $k = 3$ most frequent sequences S be:

$$C = \begin{bmatrix} 1 & 2 & 0 & 1 & 2 \\ 2 & 0 & 1 & 2 & 0 \\ 1 & 2 & 0 & 1 & 2 \\ 0 & 1 & 2 & 0 & 2 \\ 2 & 0 & 1 & 1 & 0 \end{bmatrix} \quad S = \begin{bmatrix} 1 & 2 & 0 & 1 & 2 \\ 2 & 0 & 1 & 2 & 0 \\ 0 & 1 & 2 & 0 & 2 \end{bmatrix}$$

$i = 0$
$s = 0$
$r = 0$
$S_{sr} = 1$
$i < S_{sr}$, simultaneously replace labels S_{sr} (1) and i (0) in column r (0) across S and C:

$$S[:, r] = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} \xrightarrow{\text{replace } 0 \leftrightarrow 1} \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}$$

$$C[:, r] = \begin{bmatrix} 1 \\ 2 \\ 1 \\ 0 \\ 2 \end{bmatrix} \xrightarrow{\text{replace } 0 \leftrightarrow 1} \begin{bmatrix} 0 \\ 2 \\ 0 \\ 1 \\ 2 \end{bmatrix}$$

next iteration of r ($r = 1$), $S_{sr} = 2$
$i < S_{sr}$, simultaneously replace labels S_{sr} (2) and i (0) in column r (1) across S and C:

$$S[:, 1] = \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix} \xrightarrow{\text{replace } 2 \leftrightarrow 0} \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}$$

$$C[:, 1] = \begin{bmatrix} 2 \\ 0 \\ 2 \\ 1 \\ 0 \end{bmatrix} \xrightarrow{\text{replace } 2 \leftrightarrow 0} \begin{bmatrix} 0 \\ 2 \\ 0 \\ 1 \\ 2 \end{bmatrix}$$

next iteration of r ($r = 2$), $S_{sr} = 0$
$i < S_{sr}$, S_{sr} (0) and i (0) are equal, no change in column r (2):

The next two iterations are the same to what we have already seen, so we will directly look at the result after both of them.

iterations r ($r = 3$), $S_{sr} = 1$
iterations r ($r = 4$), $S_{sr} = 2$
$i < S_{sr}$, simultaneously replace labels S_{sr} and i in column r (3,4) across S and C
This results in these matrices after the first iteration of s:

$$S \begin{bmatrix} 1 & 2 & 0 & 1 & 2 \\ 2 & 0 & 1 & 2 & 0 \\ 0 & 1 & 2 & 0 & 2 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 1 & 2 & 2 \\ 1 & 1 & 2 & 1 & 0 \end{bmatrix}$$

$$C \begin{bmatrix} 1 & 2 & 0 & 1 & 2 \\ 2 & 0 & 1 & 2 & 0 \\ 1 & 2 & 0 & 1 & 2 \\ 0 & 1 & 2 & 0 & 2 \\ 2 & 0 & 1 & 1 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 1 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 2 & 1 & 0 \\ 2 & 2 & 1 & 0 & 2 \end{bmatrix}$$

for the next iteration of $s = 1$:
$i = 1$
$s = 1$
$r = 0$
$S_{sr} = 2$

Now we repeat the previous steps for each $r = 0..4$ but in row $s = 1$
$i < S_{sr}$, simultaneously replace labels $S_{sr} = 2$ and $i = 1$ in column $r = 0$ across S and C
$i < S_{sr}$, simultaneously replace labels $S_{sr} = 2$ and $i = 1$ in column $r = 1$ across S and C
$i < S_{sr}$, $S_{sr} = 1$ and $i = 1$ are equal, no change in column $r = 2$
$i < S_{sr}$, simultaneously replace labels $S_{sr} = 2$ and $i = 1$ in column $r = 3$ across S and C
$i < S_{sr}$, simultaneously replace labels $S_{sr} = 2$ and $i = 1$ in column $r = 4$ across S and C
This results in these matrices after the second iteration of s:

$$S \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 1 & 2 & 2 \\ 1 & 1 & 2 & 1 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 0 \end{bmatrix}$$

$$C \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 1 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 2 & 1 & 0 \\ 2 & 2 & 1 & 0 & 2 \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

in the last iteration of $s = 2$:
all S_{sr} are smaller than or equal to $i = 2$, nothing changes.
This results in the final return Matrix C:

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

G Advanced Feature Outlook

As mentioned in Section 7.3, a future version of the toolbox could include an "advanced mode" that offers additional features for experienced users.
Below are a few potential features that could be included:

- **Custom Embedding Models:** Allow selection of specific embedding models, such as Sentence-BERT variants, additional external APIs, or even user-defined models, to create a better semantic representation to the dataset.
- **Language Model Selection:** Enable selection of language models for different new tasks such as keyword extraction or summarization.
- **Choose Clustering Algorithm:** Provide algorithms beyond K-means, such as DBSCAN, HDBSCAN, or hierarchical clustering, to better accommodate varying datasets.
- **Dimensionality Reduction selection:** Support selection of dimensionality reduction techniques (e.g. PCA, t-SNE, UMAP) to better match the clustering approach and visualization requirements.
- **Parameter Fine-Tuning:** Allow adjustment of algorithm parameters, such as the number of runs, initialization methods, embedding batch size, or α in the stability calculation.
- **More Export Options:** Extend export functionality to include formats like .JSON, .CSV or direct API integrations.
- **Interactive Filtering:** Add the ability to filter the displayed datapoints based on user-defined criteria, such as cluster membership, stability thresholds or sentiment.