Clustering and Sentiment Analysis of Customer NVH Feedback in the Automotive Domain

A Machine Learning Pipeline to Facilitate Extraction of Relevant Information from Large-Scale Textual Data

Master's thesis in Engineering Mathematics and Computational Science

Tomas Ekholm

Department of Mathematical Sciences

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Master's thesis in Engineering Mathematics and Computational Science 2025

Clustering and Sentiment Analysis of Customer NVH Feedback in the Automotive Domain

A Machine Learning Pipeline to Facilitate Extraction of Relevant Information from Large-Scale Textual Data

Tomas Ekholm

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2025

Clustering and Sentiment Analysis of Customer NVH Feedback in the Automotive Domain

A Machine Learning Pipeline to Facilitate Extraction of Relevant Information from Large-Scale Textual Data

Supervisor: Shubhada Deshmukh, Volvo Cars
Advisor: Yuhao Zhang, Chalmers
Examiner: Fredrik Brännström, Chalmers

Master's thesis in Engineering Mathematics and Computational Science 2025
Department of Mathematical Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2025

Clustering and Sentiment Analysis of Customer NVH Feedback in the Automotive Domain

A Machine Learning Pipeline to Facilitate Extraction of Relevant Information from Large-Scale Textual Data

Tomas Ekholm
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

Noise, Vibration, and Harshness (NVH) attributes play a significant role in shaping overall customer satisfaction with a vehicle. Automotive manufacturers often collect large volumes of textual customer feedback through surveys, offering valuable insights into how various vehicle attributes are perceived. This information is intended to help engineering teams in making informed decisions about where to focus vehicle improvement efforts. However, its unstructured nature and scale make it difficult for individual teams to extract the feedback relevant to them.

This thesis investigated the feasibility of a clustering and sentiment analysis pipeline to support NVH teams in making better use of customer feedback. The proposed pipeline combined sentence embeddings, dimensionality reduction, and clustering to group semantically similar feedback. Cluster labels were automatically generated using a large language model and manually refined when necessary. Both sentence-based and aspect-based sentiment analysis were applied to quantify sentiment and extract relevant subtopics for each cluster.

The final configuration produced 15 semantically coherent clusters from approximately 36,000 customer feedback sentences. These clusters captured distinct themes, ranging from high-level impressions of driving and ownership to specific issues regarding individual components. Sentence-level sentiment analysis successfully distinguished between positive and negative feedback showing its potential to guide improvement efforts. In contrast, aspect-based sentiment analysis was less reliable: although per-cluster aspect distributions often aligned with cluster themes, individual aspect terms were too frequently inaccurate. Nonetheless, the method shows potential, and its effectiveness could likely be substantially enhanced through domain-specific fine-tuning. Overall, the pipeline effectively facilitated the identification of relevant feedback and could aid future data-driven design and product improvement efforts.

Keywords: Clustering, Deep-Learning, HDBSCAN, K-Means, LCF-ATEPC, Machine-Learning, Natural Language Processing, PyABSA, Sentiment Analysis.

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Shubhada Deshmukh at Volvo Cars and Yuhao Zhang at Chalmers, for their guidance, support, and patience throughout this thesis. I also want to thank my examiner, Fredrik Brännström, for reviewing my work.

I am also thankful to the entire VA Insights team at Volvo for their welcoming attitude and friendly spirit. I am especially grateful to everyone I had the chance to discuss the project with, your willingness to share your thoughts made my work much easier.

Tomas Ekholm, Gothenburg, June 2025

List of Acronyms

Below is a list of acronyms that have been used throughout this thesis listed in alphabetical order:

ABSA	Aspect Based Sentiment Analysis
APC	Aspect Polarity Classification
ATE	Aspect Term Extraction
ASTE	Aspect Sentiment Triplet Extraction
BERT	Bidirectional Encoder Representations from Transformers
BiLSTM	Bidirectional Long Short-Term Memory
BoW	Bag of Words
CDM	Context-features Dynamic Mask
CDW	Context-features Dynamic Weighting
CLS	Classification token (special token in BERT used to represent entire input sequence)
DAN	Deep Averaging Network
DBCV	Density Based Cluster Validation
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
FIL	Feature Interactive Learning
GCFG	Global Context Feature Generator
HDBSCAN	Hierarchical Density-Based Spatial Clustering of Applications with Noise
LCF-ATEPC	Local Context Focus for Aspect Term Extraction and Polarity Classification
LCFG	Local Context Feature Generator
k-NN	k-Nearest Neighbor
MLM	Masked Language Modeling
MST	Minimum Spanning Tree
NLI	Natural Language Inference
NLP	Natural Language Processing
NSP	Next Sentence Prediction
NVH	Noise Vibration and Harshness
PCA	Principal Component Analysis
SBERT	Sentence BERT
SNLI	Stanford Natural Language Inference
TF-IDF	Term Frequency Inverse Document Frequency
t-SNE	t-distributed Stochastic Neighbor Embedding
UMAP	Uniform Manifold Approximation and Projection
USE	Universal Sentence Encoder

Contents

List of Acronyms **ix**

1 Introduction **1**
1.1 Background . . . 1
1.2 Research Questions and Limitations . . . 2
1.3 Confidentiality . . . 2

2 Theory **3**
2.1 Text Vectorization Techniques . . . 3
2.1.1 Count-Based Methods . . . 3
2.1.2 Static Word Embeddings . . . 4
2.1.3 Contextualized Word and Sentence Embeddings . . . 4
2.2 Challenges of High-Dimensional Data . . . 6
2.3 Discovering Structure in Text: Clustering . . . 8
2.3.1 K-Means . . . 9
2.3.2 HDBSCAN . . . 10
2.3.3 Clustering Evaluation Metrics . . . 11
2.4 Opinion Mining in Text . . . 16
2.4.1 Sentiment Analysis: A Brief Introduction . . . 16
2.4.2 LCF-ATEPC Model . . . 16

3 Methodology **19**
3.1 Dataset and Preprocessing . . . 19
3.2 Generating Sentence Embeddings . . . 20
3.3 Dimensionality Reduction . . . 21
3.4 Clustering Strategy . . . 21
3.4.1 Cluster Evaluation . . . 22
3.4.2 Experimental Setup for Hyperparameter Evaluation . . . 23
3.4.3 Selecting the Final Clustering Configuration . . . 25
3.5 Sentiment Extraction . . . 26

4 Results and Analysis **27**
4.1 Final Clustering Model . . . 27
4.1.1 Parameters and Model Choice . . . 27
4.1.2 Results on Test-set . . . 28
4.1.3 Sentiment Trends . . . 29
4.2 Comparing K-Means and HDBSCAN Clusterings Qualitatively . . . 30

4.3 Results from Hyperparameter Evaluation Experiments 33

5 Conclusions **37**
5.1 Addressing the Research Questions 37
5.2 Limitations and Future Work 38

Bibliography **41**

A Appendix 1 **I**
A.1 BERTopic Configuration Settings I
A.2 The Results for Experiment 3: Effect of Dimensionality and k on K-Means Cluster Quality II

1
Introduction

This thesis, conducted in collaboration with Volvo Cars, explores the effectiveness of machine learning methods for clustering and sentiment analysis of consumer feedback related to Noise, Vibration, and Harshness (NVH) in the automotive domain. The objective was threefold: (1) to uncover latent structure in unstructured textual customer feedback through clustering, (2) to assess whether sentiment analysis at the cluster level can reveal useful insights into how customers perceive different vehicle attributes, and (3) to explore whether applying aspect-based sentiment analysis (ABSA) within those clusters can provide more specific information about which product features customers are commenting on and how they feel about them.

1.1 Background

NVH attributes are key contributors to perceived vehicle quality and the overall driving experience. Factors such as wind noise, road-induced vibration, and brake performance directly impact customer satisfaction. Beyond comfort and quality, NVH also has safety implications. According to [AA15], low-frequency road-noise can impair driving performance and induce driver sleepiness, a main contributory factor in crashes. As a result, NVH characteristics are typically ranked among the top five most important attributes in vehicle design [QAPS09]. For automotive manufacturers, especially in a competitive market, the ability to identify and address NVH-related concerns through customer feedback is increasingly important. By analyzing such feedback, automakers can pinpoint specific design areas in need of improvement, gaining a valuable competitive advantage.

Volvo Cars collects large amounts of customer feedback through regular customer surveys. While this data presents valuable opportunities to gain insights into customer sentiments, the fact that textual data is inherently unstructured, combined with the sheer volume of responses, makes it challenging to interpret. Manually analyzing thousands of responses is very time-consuming, limiting the ability to extract meaningful information. To address this challenge, an option is to use machine learning techniques that can process large amounts of data at scale, such as clustering and large language models (LLMs). Clustering enables the automatic grouping of semantically similar feedback, while LLMs provide powerful capabilities for analyzing human language through their ability to capture semantic context. Together, these techniques support automated sentiment analysis [WM22, YZL23], topic modeling [Gro22], and large-scale clustering of consumer feedback [ATOS23].

This thesis investigates the feasibility of applying machine learning techniques to structure and analyze consumer feedback related to NVH attributes. To this end, a variety of machine learning and deep learning models are used. Sentence-BERT (SBERT), a model based on Bidirectional Encoder Representations from Transformers (BERT), a transformer architecture known for producing context-aware language representations, is used to transform textual data into a numerical format necessary for clustering. Uniform Manifold Approximation and Projection (UMAP) is then applied to reduce these high-dimensional embeddings into a lower-dimensional space that is more suitable for clustering. Both K-Means and Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) are then explored for grouping semantically similar responses. Finally, sentiment analysis is carried out using models from the PyABSA framework, which includes BERT-based models specifically adapted and trained for sentiment analysis tasks. The goal is to assess whether these techniques can effectively organize feedback around meaningful topics and extract sentiment trends associated with them, contributing to a more systematic and scalable approach to understanding customer experience.

1.2 Research Questions and Limitations

Based on the objective of this thesis, the following research questions were posed:

- Can consumer feedback be clustered in such a way that each cluster aligns with specific NVH related topics handled by different teams within the NVH department at Volvo? If not, do other meaningful groupings emerge that still provide value?
- Can sentiment analysis applied to these clusters yield insights that help in understanding general customer opinions about specific aspects of the vehicle?
  - For example, can patterns in sentiment relating to brake performance, wind noise, vibrations, or other attributes relevant to the NVH department be extracted?

These questions were explored with certain limitations. The sentiment analysis models were not formally evaluated beyond brief result inspection, so definitive conclusions about their accuracy cannot be drawn. However, their outputs can still roughly indicate clustering validity. For instance, by assessing whether the identified aspects and sentiments match with the theme and content of discovered clusters.

1.3 Confidentiality

The data used in this master's thesis is considered private customer information and is subject to a confidentiality agreement. Therefore, all examples of customer responses are fictitious to protect customer confidentiality. Additionally, any extracted aspect terms from clusters or automatically generated cluster labels have been anonymized in accordance with the company's request.

2
Theory

This chapter presents the theoretical background supporting the methods used in this thesis. It begins by explaining how text can be represented as numerical data, from early frequency-based models to modern deep learning techniques for contextual embeddings. It then outlines the challenges of high-dimensional data, particularly for distance-based clustering methods and metrics. As a solution, UMAP is introduced for dimensionality reduction. The chapter continues with an overview of clustering as a way to uncover structure in text, covering the K-Means and HDBSCAN algorithms, along with relevant evaluation metrics. It concludes with a brief introduction to sentiment analysis and a description of the Local Context Focus for Aspect Term Extraction and Polarity Classification (LCF-ATEPC) model.

2.1 Text Vectorization Techniques

In order to perform clustering or sentiment analysis on textual data, the text must first be converted into a numerical format that machine learning algorithms can process. Since these algorithms operate on numerical data, this transformation is a critical first step. By representing text as vectors in a vector space we can group those that are nearby in this space together. But for the clustering to be meaningful to us, the vector representations should also have the property that semantically similar texts yield vectors that lie in close proximity of each other in this space.

2.1.1 Count-Based Methods

Early methods for this conversion is the lexicographical Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF) methods. BoW only considers the frequency of words within a text, while TF-IDF further builds upon this by considering the frequency of words in a document against the frequency of words in the entire corpus[1]. These methods are however very rudimentary and have several drawbacks such as, ignoring word ordering, making it impossible to capture context and word relationships [JMN23]. The resulting embeddings are also extremely high dimensional as the dimensionality equals the vocabulary size of the corpus. Furthermore these models cannot capture semantic meaning of related but different words, such as *cat* and *kitten* as a result of only focusing on word frequency [JMN23].

[1]In the context of natural language processing (NLP), a *document* refers to the basic unit of text being processed, which can range from a full book to a single sentence or phrase depending on the task. A *corpus* is the collection of all such documents.

2.1.2 Static Word Embeddings

To overcome the limitations of frequency-based methods, word embedding models such as Word2Vec, GloVe, and FastText were developed [ZPS+24]. These models represent each word as a vector in a dense vector-space of significantly lower dimensionality than BoW or TF-IDF. The main idea is that words with similar meanings and that appear in the same context will have vectors that lie close together, effectively capturing semantic relationships. Word2Vec and FastText are based on shallow neural network architectures trained to predict a target word from its context, or vice versa. The key difference is that Word2Vec treats full words as the unit of representation, while FastText incorporates subword information by using character-level n-grams, which helps in representing rare or misspelled words. Glove on the other hand works by constructing a word co-occurrence matrix that records how often words appear together in a corpus, then implicitly factorizes this matrix by optimizing a function that models these co-occurences, effectively generating word vectors that capture these patterns.

These embedding models all capture general semantic relationships between words allowing them to better represent language than count-based methods of text representation. Despite these improvements, some limitations still remain. Even if the vector embeddings are of significantly lower dimensionality than those generated by BoW and TF-IDF, they can still be considered high-dimensional for the purpose of clustering. More importantly, these models produce static representations, meaning that each word receives the same vector regardless of context, even when the word has multiple meanings. For example, they cannot distinguish between *apple* in *apple pie* and *Apple* as in *Apple computer* [ZPS+24].

2.1.3 Contextualized Word and Sentence Embeddings

InferSent and USE

This introduces the need for context sensitive representations, which adjust a word's representation based on the context in which it appears. Several models have been proposed to address this, including InferSent [CKS+18] and the Universal Sentence Encoder (USE) [CYyK+18], which both produce fixed-length sentence embeddings designed to capture the meaning of entire sentences.

USE generates sentence embeddings using either a transformer based architecture or a Deep Averaging Network (DAN). The model is trained using multi-task learning on a mix of supervised and unsupervised tasks, such as sentence prediction, conversational response selection, and Natural Language Inference (NLI). USE embeddings have demonstrated strong performance on semantic textual similarity (STS) benchmarks such as SemEval-2017 Task 1 [CDA+17] and can be fine-tuned for improved results in specific applications [CYyK+18].

InferSent is a sentence encoder based on the Bidirectional Long Short-Term Memory (BiLSTM) architecture, pre-trained on NLI tasks with the goal of learning univer-

sal representations of sentences that then can be used for other tasks. By utilizing a BiLSTM architecture, InferSent can capture context both from past and future words within a sentence, giving more accurate embeddings. Training on NLI tasks also allows InferSent to learn sentence representations that generalize well across different NLP applications. This is because it is assumed that NLI tasks require a high level understanding of language to be successfully completed. For this purpose the Stanford Natural Language Inference (SNLI) dataset [BAPM15] was used for training [CKS+18], which provides a large collection of sentence pairs annotated for natural language inference, requiring models to determine whether a given hypothesis logically follows from, contradicts, or is unrelated to a premise.

BERT

A major breakthrough in NLP was the introduction of Bidirectional Encoder Representations from Transformers (BERT) model [DCLT19]. BERT uses the encoder part of the Transformer architecture to process input text bidirectionally through self-attention, a mechanism that allows the model to consider both the left and right contexts of each word[2] and to weigh the importance of surrounding words when computing its representation. This enables BERT to produce contextualized word embeddings that effectively capture semantic relationships in text. BERT is pre-trained on two tasks, Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). In MLM, some of the input words are randomly masked, and the model learns to predict the masked words from the context alone. NSP involves predicting whether one sentence follows another, allowing the model to learn inter-sentence relationships [Ala19].

While BERT is highly effective for many NLP tasks, it was not designed to produce embeddings for full sentences. To represent an entire sentence as a single vector, common workarounds include averaging or pooling the word embeddings, or use the classification (CLS) token[3] output to produce a fixed length sentence embedding. However, these methods are ill-suited to preserve the semantic meaning of sentences, leading to subpar performance on tasks involving semantic similarity.

BERT can also directly compare the semantic similarity between two sentences internally by processing them as a sentence pair, thus avoiding the performance ruining step of averaging, pooling or using the CLS token. The drawback of this is that it is extremely inefficient for comparing many sentences. To compare n sentences, BERT must encode every possible sentence pair together, resulting in a computational complexity of $\frac{(n-1)\cdot n}{2}$ for the number of forward passes through the model. This makes BERT impractical for applications that requires the comparison of large numbers of sentences [RG19].

[2]Strictly speaking, BERT operates on tokens, which may be whole words or subword units, but for simplicity the term "word" is used in the text.

[3]A token added to the beginning of every input sequence. Its final representation is used to summarize the entire sequence for classification tasks.

SBERT

Building upon the BERT architecture but adapting it for the explicit task of generating sentence embeddings, SBERT was introduced in 2019 [RG19]. Unlike vanilla BERT, which is not optimized for producing comparable sentence-level representations, SBERT employs a Siamese network structure in which two identical BERT models with shared weights encode separate input sentences. This design enables efficient training on sentence similarity tasks such as STS and NLI using paired sentences as input. Training on these tasks is essential for the model to learn to produce semantically meaningful sentence embeddings.

During training each sentence is passed through its own separate BERT model, and the resulting word-level embeddings are aggregated using a pooling operation. Because SBERT is explicitly trained on sentence similarity tasks, this pooling step is now optimized to produce sentence embeddings for sentence similarity tasks, unlike in vanilla BERT where no such optimization is performed.

Once trained, SBERT allows for efficient precomputation of sentence embeddings drastically reducing the computational cost for sentence similarity tasks compared to BERT. For example, while BERT would require approximately 50 million forward passes to find the most similar pair in a set of 10,000 sentences using pairwise comparisons, SBERT only requires 10,000 forward passes to encode the sentences, after which comparisons can be efficiently made in the embedding space using, for example cosine similarity. In addition to being significantly more efficient than BERT for similarity tasks, SBERT also outperforms earlier sentence embedding models such as InferSent and the USE on a range of benchmark tasks [RG19].

2.2 Challenges of High-Dimensional Data

Advances in text representation have significantly reduced the dimensionality of embedding, from being in the tens of thousands to just a few hundred (for some models still in the thousands). However, sentence embeddings with a dimensionality in the hundreds, such as those produced by SBERT, can still be considered high-dimensional for the purposes of clustering.

Why is this bad? The issue lies in the so called *curse of dimensionality*. As the number of dimensions increases, the distance between any two random points converges towards the average distance, meaning all points appear nearly equidistant [PGW25]. This reduces the contrast between dense and sparse regions in the data, which becomes a significant problem for clustering algorithms that rely on density measures, such as HDBSCAN. In such cases, the algorithm struggles with detecting meaningful clusters. For example, according to the HDBSCAN documentation, the algorithm is generally effective up to around 50 to 100 dimensions, after which performance tends to degrade significantly [MHA25a].

Not only is the performance of clustering algorithms impacted by the curse of di-

mensionality, it also affects the evaluation of clustering quality. Most intrinsic clustering metrics rely on meaningful distance calculations between points and clusters. However, as dimensionality increases, these distances become less informative. For example, several intrinsic metrics, including widely used ones like the Silhouette Score and the Davies-Bouldin Index, have been shown to produce artificially inflated scores in high-dimensional spaces [TR16]. This effect not only distorts the interpretation of clustering results, it also limits the comparability of cluster quality across embedding spaces of different dimensionality. Due to these challenges, it is often necessary to reduce the dimensionality of sentence embeddings prior to clustering. To address this, UMAP, a dimensionality reduction technique, is introduced as a suitable solution.

UMAP

UMAP is a dimensionality reduction technique, that reduces the dimensionality of a dataset by finding a low-dimensional projection that preserves the topological structure of the original high-dimensional space [McI25]. This is achieved by first estimating the local relationships between points in the original high-dimensional space, then randomly initializing corresponding points in a lower-dimensional space. These points are iteratively rearranged so that points that were close in the high-dimensional space become close in the reduced space. The UMAP algorithm consists of two main steps:

1. Graph construction: A weighted k-Nearest Neighbor (k-NN) graph representation of the original high-dimensional data is created. In this graph each data point is a vertex, and the edge weights are a measure of similarity between data points.
2. Graph optimization: A corresponding graph in a reduced space is found by minimizing a cost function that reflects the difference between similarities of points in the two different spaces, this function is minimized by adjusting the position of the projected points.

The algorithm has two major parameters which are `n_neighbors` which sets the number of neighbors to consider in the graph creation step, and `n_components` which sets the dimensionality of the reduced space.

Mathematically it can be formulated in the following way: Given a dataset

$$X = [x_1, ..., x_n] \in \mathbb{R}^{d \times n}$$

where n is the number of data points and d is the dimensionality of the data. UMAP first constructs a k-NN graph where the k nearest neighbors of x_i is defined as:

$$N_i := \{x_{i,1}..., x_{i,k}\}$$

Next, the neighbor relationship is modeled in a stochastic manner using a Radial Basis Function kernel for the measure of similarity between points. The probability

of a point x_i having point x_j as neighbor is then:

$$p_{i|j} := \begin{cases} \exp(-\frac{||x_i - x_j||_2 - \rho_i}{\sigma_i}) & \text{if } x_j \in N_i \\ 0 & \text{otherwise} \end{cases}$$

where ρ_i is the distance from x_i to its nearest neighbor; σ_i is a scaling parameter, calculated so that the total similarity of x_i to its k nearest neighbors is normalized to $log_2(k)$. A symmetric similarity measure with respect to i and j can then be defined as:

$$p_{ij} := (p_{i|j} + p_{j|i} - p_{i|j} \cdot p_{j|i})$$

Let the reduced space we are projecting our high-dimensional data onto be:

$$Y = [y_1, ..., y_n] \in \mathbb{R}^{p \times n}$$

where p is the dimension of our reduced space, ($p \ll d$), and the points $y_1, ..., y_n$ are randomly initialized.

In the reduced space, the similarity of points y_i and y_j, which also represents the probability they are neighbors, is defined as:

$$q_{ij} := (1 + a||y_i - y_j||_2^{2b})^{-1}$$

where a and b are minor hyperparameters that UMAP's documentation recommend to be left at their default values: $a \approx 1.929$, $b \approx 0.7915$.

The goal is then to minimize the difference between the similarities of points in the high-dimensional space and those in the reduced-space. This is done by minimizing the following cost function w.r.t y using stochastic gradient descent:

$$\min_{\{y_i\}_{i=1}^n} -\sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \left(p_{ij} \ln(q_{ij}) + (1 - p_{ij}) \ln(1 - q_{ij}) \right)$$

Further details about the optimization procedure can be found in the original UMAP paper [MHM20] or in this tutorial survey paper [GGKC21]. Although absolute distances between points in the reduced space lack meaning and direct connection to their high-dimensional counterparts, UMAP preserves both local and global structure effectively, making it well suited for tasks such as clustering.

2.3 Discovering Structure in Text: Clustering

Given a large set of unstructured data, one often wants to impose some structure on it, organizing it in a way that makes it easier for a human to understand and interpret. Clustering is an unsupervised machine-learning technique used to discover latent patterns in data by grouping similar observations together, thereby creating structure within otherwise unstructured data [ATOS23]. It is a general-purpose tool used across many domains for tasks such as customer segmentation, anomaly

detection or image segmentation. In this thesis, the goal is to discover patterns and themes in customer opinions. This means grouping sentences that are semantically similar into coherent clusters, which each should represent a common topic or theme. These clusters can then form the basis for further analysis. An implication of working with textual data is however that the usefulness of any kind of clustering will strongly depend on the quality of the underlying text-representations and how well they manage to capture semantic meaning.

2.3.1 K-Means

K-Means clustering is a simple but efficient clustering method where the objective is to partition the data into K clusters $C = \{C_1, C_2, ..., C_K\}$ so that the within-cluster sum of squares (WCSS) is minimized [IEA+23]. It has only one parameter to tune and that is the number of clusters K. While K-Means is computationally efficient and easy to implement, it makes several key assumptions about the structure of the data. Most notably, it assumes that clusters are spherical (or convex), of roughly equal size, and with similar densities. When these assumptions are violated, the algorithm's performance can degrade significantly. K-Means can be mathematically formulated as the following optimization problem:

$$\arg\min_{C} \sum_{j=1}^{K} \sum_{x_i \in C_j} ||x_i - \mu_j||^2$$

where C is the partitioning of the data; K is the number of clusters; C_j is the set of points assigned to cluster j; x_i is a data point; and μ_j is the centroid of cluster k. It is algorithmically implemented as follows:

Algorithm 1 K-Means algorithm (classical implementation [IEA+23])

1: Choose the number of clusters K
2: Randomly initialize each cluster centroid $\mu_1, \mu_2, ..., \mu_K$
3: **while** centroids have not converged **do**
4: Assign each data point x_i to the nearest centroid:

$$C_m = \{x_i : ||x_i - \mu_m||^2 \leq ||x_i - \mu_j||^2, \forall j \ 1 \leq j \leq K\}$$

5: Update each centroid μ_j as the mean of assigned points:

$$\mu_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

6: **end while**
7: Return clusters and centroids

2.3.2 HDBSCAN

HDBSCAN is an extension of Density-Based Spatial Clustering of Applications with Noise (DBSCAN) that introduces a hierarchical component. Instead of producing a single clustering, it builds a hierarchy of clusters based on varying density thresholds, and then extracts the most stable flat clustering from this hierarchy [CMS13].

The algorithm can be broken down into five main steps [MHA25b]:

1. Transform the space based on density using a modified distance metric.
2. Build a minimum-spanning tree (MST) from a distance-weighted graph.
3. Construct a hierarchy of clusters from the MST.
4. Condense the cluster hierarchy using the `minimum_cluster_size` parameter.
5. Extract the most stable clustering from the condensed tree.

The first step consists of calculating the mutual reachability distance between all points. This distance modifies the original distance metric (typically Euclidean) by incorporating local density estimates for the points being compared. This local density estimate around a point is called the core distance, and is defined as the distance to its k-nearest neighbor, where k is set by the `min_samples` parameter. The mutual reachability distance between two points a and b is then given by:

$$d_{mreach-k}(a, b) = \max(core_k(a), core_k(b), d(a, b))$$

where $core_k(a)$ is the distance from a to its k-th nearest neighbor; $core_k(b)$ is the distance from b to its k-th nearest neighbor; $d(a, b)$ is the distance between points a and b in the original metric. With mutual reachability distance, points in dense regions with low core distance, keeps their distance to other points unchanged, while sparser points are pushed away to be atleast their core distance away from any other points.

In the second step, the data is treated as a weighted graph where each data point is a vertex, and the edge weights correspond to the mutual reachability distances between points. A minimum spanning tree[4] is then generated from this graph using Prim's algorithm.

With the data represented as a minimum spanning tree, it is possible to convert that into a hierarchy of connected components. This is done by first sorting the edges of the tree in ascending order by distance. All points are initially treated as their own clusters, and the algorithm then iterates through the edges, merging the clusters at the endpoints of each edge. This yields a hierarchy of clusters by mutual reachability distance.

The fourth step is to condense the cluster-hierarchy tree into a smaller simplified tree. This is done using the `min_cluster_size` parameter. The algorithm

[4] A minimum spanning tree (MST) is a subset of the edges in a connected, weighted graph that connects all the vertices together without any cycles and with the minimum possible total edge weight.

iterates through the cluster-hierarchy in a top-down manner and, at each split, checks whether either of the resulting clusters created by the split is smaller than `min_cluster_size`. If one of them is too small, it is not considered a true cluster split, but instead, the larger part is considered to be the same cluster as before, while the smaller is regarded as points falling out of the cluster. If both resulting clusters are large enough then it is considered a true cluster split, and the split remains. This creates a much simpler condensed cluster hierarchy tree, with much fewer splits.

Lastly, a flat clustering is extracted from the condensed tree. The default method (‘*eom*‘) selects the most stable set of clusters from the hierarchy. Here, stability is computed individually for each cluster and refers to how long that cluster persists as a coherent group throughout the condensed cluster hierarchy tree. Stability is defined as:

$$\text{Stability} = \sum_{p \in cluster} (\lambda_p - \lambda_{birth})$$

$\lambda = \frac{1}{d_{\text{mreach-k}}}$: a density measure, inversely proportional to the mutual reachability distance.
λ_{birth}: the λ value where there cluster split of from its parent cluster.
λ_{death}: the λ value where cluster splits into smaller clusters.
$\lambda_p \in [\lambda_{birth}, \lambda_{death}]$: For a point p in a given cluster, this is the value where this points fall out of the cluster.

The algorithm for selecting the most stable clusters proceeds by traversing the condensed tree from the bottom up, starting by considering each leaf node to be a selected cluster. At each internal node it passes on the way up, it compares the clusters own stability against the combined stability of its child clusters. If the combined stability of the child clusters exceeds that of the parent cluster, the cluster stability is updated to match the sum of its children's stabilities. If the opposite is true, declare the cluster to be a “selected cluster” and deselect all its children. After arriving at the root node, the current set of selected clusters becomes the returned flat clustering. There is also an additional optional parameter, `cluster_selection_epsilon` that can influence the final clustering by merging clusters whose distance of separation is less than this value [MHA25c].

One of the main strengths of HDBSCAN is that it makes no assumptions about the underlying shape, size, or density of clusters. This flexibility makes it especially well-suited for real-world data, where such idealized assumptions often do not hold. Additionally, HDBSCAN can identify and label points as noise further enhancing its robustness in practice.

2.3.3 Clustering Evaluation Metrics

To evaluate and compare different clustering results, appropriate clustering metrics are required. These metrics quantify clustering quality by measuring factors such as cluster compactness and separation, common indicators of effective clustering. It can then be an idea to try and optimize the clustering with these scores as an

objective function to find as compact and well-separated clusters as possible.

However, in the context of language, some variation within a cluster might be worth preserving as the same topic often can be described in a wide variety of ways. It is also possible that textual data exhibits a latent hierarchical structure, allowing for clustering at different levels of abstraction. Where a result yielding few large clusters may represent broad themes, while another clustering giving many small cluster captures more specific subtopics. All of these clusterings can potentially reflect meaningful structure.

This implies that there most likely does not exist one single optimal correct clustering, and that more compact clusters are not necessarily more meaningful. Clustering metrics should therefore be seen more as a form of guidance than an absolute truth. They are however still essential for efficiently comparing multiple clusterings and filtering out poor results, but it remains important to manually inspect clusterings to determine whether they make intuitive sense as well. Below follows several commonly used metrics for measuring cluster quality.

Silhouette Score

The Silhouette score measures how similar a point is to its own cluster compared to other clusters [Des16]. The Silhouette Score ranges from -1 to +1, where a score of +1 indicates the best possible clustering. If $s(i)$ for a single point i is negative, it suggests that the point might be assigned to the wrong cluster. A score close to 0 means the point lies near the boundary between two clusters, and a score close to +1 indicates that the point is well-clustered. It is defined as:

$$S = \frac{1}{n} \sum_{i=1}^{n} s(i)$$

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

where S is the mean Silhouette Score of all points; $s(i)$ is the Silhouette Score for a single data point; $a(i)$ is the average distance from point i to all other points in the same cluster; $b(i)$ is the average distance from point i to all points in the nearest cluster; n is the total number of data points.

Davies-Bouldin Index (DB)

The DB index evaluates clustering based on the average similarity between each cluster and its most similar counterpart [Des16]. The DB index is considered better when it is lower, as it indicates that clusters are either highly compact or their centroids are well-separated. In the ideal case, a perfect possible clustering would yield an index of 0. It is defined as:

$$DB = \frac{1}{K} \sum_{i=1}^{K} \max_{j \neq i} \left(\frac{s_i + s_j}{d(i,j)} \right)$$

where K is the number of clusters; s_i is the average distance between points in cluster i and its centroid; $d(i,j)$ is the distance between the centroids of clusters i and j; The expression $\max_{j \neq i} \left(\frac{s_i + s_j}{d(i,j)} \right)$ identifies the most similar (least well-separated) cluster j to i, by taking the worst-case ratio between within-cluster scatter and between-cluster distance.

Calinski-Harabasz Index (CH)

The CH index, or sometimes Variance Ratio Criterion measures the quality of clustering based on the ratio of the sum of between-cluster dispersion (BCSS) to within-cluster dispersion (WCSS) [Des16]. The CH index is considered better when it is higher as it indicates that the clusters are well-separated, the dispersion between clusters is large, and the clusters are compact. There is no upper bound for this Index. It is defined as:

$$CH = \frac{N - K}{K - 1} \cdot \frac{BCSS}{WCSS}$$

$$BCSS = \sum_{i=1}^{K} n_i \cdot \|\mathbf{c}_i - \mathbf{c}\|^2$$

$$WCSS = \sum_{i=1}^{K} \sum_{x_j \in C_i} \|x_j - \mathbf{c}_i\|^2$$

where N is the total number of data points; K is the number of clusters; n_i is the number of points in the i-th cluster; $\mathbf{c}_i$ is the centroid of the i-th cluster; $\mathbf{c}$ is the overall centroid of the entire dataset; C_i is the i-th cluster; x_j represents a data point in cluster C_i

Density-Based Cluster Validation (DBCV)

DBCV is a metric that evaluates the quality of clustering based on the density within clusters and the density separation between them. A good clustering should have dense clusters separated by low density regions [MJC+]. The DBCV index ranges from -1 to +1, where higher values indicate well-separated, dense clusters, while lower values suggest overlapping clusters with low density. The implementation used in this thesis (available at: https://github.com/christopherjenness/DBCV/) is a slightly modified version of the original DBCV formulation, but its core idea remains the same and can be expressed mathematically as follows:

$$\text{DBCV} = \sum_{i=1}^{K} \frac{|C_i|}{N} \cdot \text{Validity}(C_i)$$

where K is the number of clusters; N is the total number of observations; $\text{Validity}(C_i)$ is the validity of cluster C_i defined as follows:

$$\text{Validity(C)} = \frac{DSPC(C, C_{nearest}) - DSC(C)}{\max\{DSPC(C, C_{nearest}),\ DSC(C)\}}$$

where DSC(C) is the density sparseness of a cluster, defined as the maximum edge weight of the internal edges of the MST of cluster C; DSPC(C_i, C_j) is the density separation of a pair of clusters, defined as the minimum reachability distance between the internal nodes of the MSTs of clusters C_i and C_j; MST(C) is the minimum spanning tree of cluster C based on mutual reachability distance.

The mutual reachability distance[5] between two points a and b is given by:

$$\text{mrd}(a, b) = \max\{\text{core}(a), \text{core}(b), d(a, b)\}$$

where $core(\cdot)$ is the density estimate around a point, based on the inverse of the sum to the points k-nearest neighbors; $d(a, b)$ is the distance between points a and b in the original metric.

Limitations of Selected Intrinsic Metrics

While useful, the intrinsic metrics selected come with important limitations. As discussed in Section 2.2, the curse of dimensionality can cause metrics like the Silhouette Score and DB index to yield artificially inflated scores, as distance-based measures become less reliable in high-dimensional spaces [TR16]. Although the CH index is generally more robust, it is still influenced by how UMAP reshapes the embedding space (see Section 3.3). Therefore, these metrics should not be used to compare clustering results across different UMAP projections, but they remain valid for comparisons within the same reduced space.

Cluster Prediction Strength

Cluster prediction strength is a metric that measures the stability of a clustering by assessing how consistently pairs of points occur in the same cluster when clustered by clusterers trained on different subsets of the same dataset [Ta05]. Since it only considers whether points are assigned to the same cluster over different clusterings, and not how closely they are arranged in space, it is robust to the effects of high dimensionality and remains applicable even when distance metrics lack meaningful interpretation. It is often used to identify the optimal number of clusters, as true clusterings are assumed to be stable. The metric ranges from 0 to 1 where a value of 1 means the clustering remain constant.

A heuristic for finding the optimal number of clusters, is to select the maximum cluster size for which the prediction strength is above a set threshold, usually 0.9 or 0.8 if the clusters are considered to be well separated. It is also suggested to use cross-validation over a number of folds and taking the average to get a more stable estimate [Ta05].

[5] This is not exactly the same mutual reachability distance as used in the HDBSCAN implementation due to a different core distance function, hence the different notation.

Algorithm 2 Cluster Prediction Strength

1: Split the data into training and test sets: X_{train} and X_{test}
2: Cluster the two sets separately but using the same algorithm and settings.
3: Create a co-membership matrix $D[C(X_{train}, k), X_{test}]$ where $C(X_{train}, k)$ is the cluster algorithm fitted to the training set, k is the number of clusters.
4: Set elements in the co-membership matrix with index i, i' to 1 if observations i and i' in the test set are assigned to the same cluster, and 0 otherwise.
5: Measure how well training set centroids predict co-membership in the testset. This is done by calculating the proportion of observations in the test set that are assigned to the same test cluster, that also are assigned to the same cluster based on the training set centroids.
6: Compute the prediction strength for $C(\cdot, k)$ as:

$$ps(k) = \min_{1 \leq j \leq k} \frac{1}{n_{kj}(n_{kj} - 1)} \sum_{i \neq i' \in A_{kj}} D[C(X_{train}, k), X_{test}]_{ii'}$$

where n_{kj} is the number of observations in cluster j; A_{kj} is the indices of the observations in test-cluster j. That is, the smallest proportion of observations that belong to the same cluster in both the test clusters and using the training centroids of all the k test clusters is the prediction strength for a given k.

Classification Metrics

If ground-truth labels exist for the data, and each cluster label can be mapped to one of these labels, classification metrics can be used to assess cluster correctness as well.

Accuracy measures the total number of correct classifications.

$$\text{Accuracy} = \frac{\text{True Positives}_i + \text{True Negatives}_i}{N}$$

where N is the total number of observations.

Class precision measures the accuracy of positive predictions for each class i.

$$\text{Precision}_i = \frac{\text{True Positives}_i}{\text{True Positives}_i + \text{False Positives}_i}$$

Class recall, or sensitivity, measures the ability to identify all relevant instances of a class.

$$\text{Recall}_i = \frac{\text{True Positives}_i}{\text{True Positives}_i + \text{False Negatives}_i}$$

The F1-score is the harmonic mean of precision and recall, providing a balance between the two metrics. The weighted F1-score averages the F1-scores for each class, weighted by class frequency, making it suitable for imbalanced datasets:

$$\text{Weighted-F1} = \sum_{i=1}^{K} \frac{n_i}{N} \cdot \frac{2 \cdot \text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}$$

where K is the number of classes (clusters); n_i is the number of true observations of class i; N is the total number of observations; Precision_i and Recall_i are the precision and recall for class i.

2.4 Opinion Mining in Text

2.4.1 Sentiment Analysis: A Brief Introduction

Opinion mining, also known as sentiment analysis, is a subfield of Natural Language Processing (NLP) that focuses on extracting and analyzing subjective information such as opinions, attitudes, and sentiments expressed in text. The goal is to understand how people feel about specific entities or topics.

In both traditional Sentiment Analysis (SA) and Aspect-Based Sentiment Analysis (ABSA), an entity refers to a specific object of interest, such as a product, service, company, individual, or issue, that people express opinions about. An aspect is a particular attribute, component, or feature of that entity. The sentiment represents the subjective attitude (positive, negative, or neutral) toward either the entity as a whole or one of its specific aspects. For example, in the sentence "*I like the sound system but the design is ugly*", the implied entity is a vehicle. The aspects mentioned are "*sound system*" and "*design*", and these are associated with the opinion expressions "*like*" (positive) and "*ugly*" (negative), respectively. Sentiment analysis tasks are typically categorized into the following levels [WKR24]:

- **Document-Level Sentiment Analysis:** This is where one tries to determine if the overall opinion of a complete document is positive, negative or neutral. In this case, a central assumption is that only opinions regarding one entity is expressed per document.
- **Sentence-Level Sentiment Analysis:** In this approach, the task is to classify the sentiment of individual sentences. The assumption here being that only opinions regarding one entity is expressed per sentence.
- **Aspect-Based Sentiment Analysis (ABSA):** This is the most fine-grained approach and seeks to identify sentiments towards specific aspects or features of a given entity. This allows multiple opinions regarding different aspects of an entity to be expressed in every sentence. The main subtasks of any ABSA approach are aspect term extraction (ATE) and aspect polarity classification (APC) [ZLD+22].

2.4.2 LCF-ATEPC Model

Local Context Focus for Aspect Term Extraction and Polarity Classification (LCF-ATEPC) is a multilingual, multi-task model for extracting aspect terms and determining their polarity. A key features of this model is that it uses multi-task learning

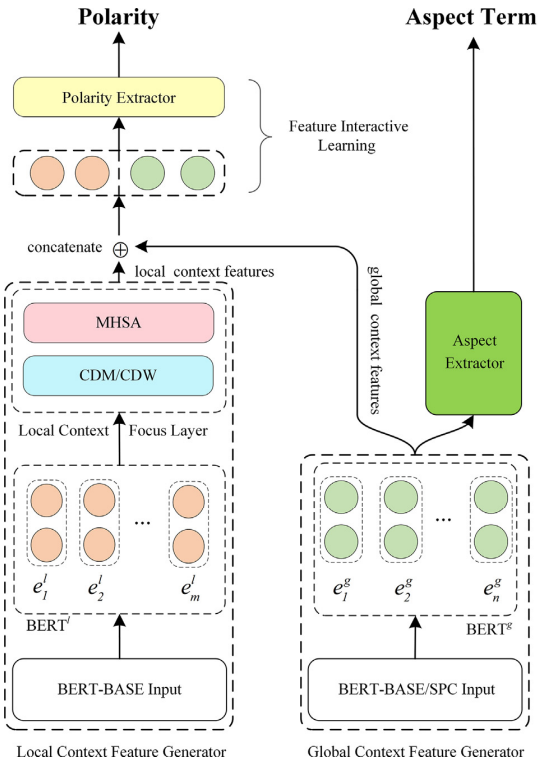


Figure 2.1: Multitask Learning Framework for LCF-ATEPC. Figure reused from YangHeng, https://github.com/yangheng95/LCF-ATEPC, GitHub (2019), licensed under the MIT License.

to learn how to perform both sub-tasks of ATE & APC in parallel, in contrast to many other ATEPC models which tends to learn each task in sequence. Like many other recent approaches, LCF-ATEPC is built on top of BERT and leverages its multi-head self-attention mechanism to generate contextual embeddings. In addition, it also incorporates a local context focus (LCF) mechanism that emphasizes the context surrounding each aspect.

The Model Architecture

The architecture consists of two main modules, the local context feature generator (LCFG) and the global context feature generator (GCFG), see Figure 2.1. Each module is based around their own instance of a pretrained BERT model to encode the input sentence [YZY+21]. It is worth noting that this is different from SBERT which uses a siamese structure where the models share the weights. A key aspect of the architecture is also that during training, each word is assigned both an aspect term label and a sentiment polarity label, enabling the model to optimize for aspect term extraction and polarity classification in parallel.

The GCFG component processes the entire input sentence to learn global context features through its internal BERT instance. In parallel, the LCFG focuses on the local context surrounding each target aspect by first encoding the sentence through

its own BERT instance, and then feeding the output through the LCF layer. This LCF layer emphasize features that belong to the local context of a target aspect by using either a context-features dynamic mask (CDM) or context-features dynamic weighting (CDW) layer. Both approaches rely on semantic-relative distance (SRD) which measures how far a token is from the target aspect, to either mask non-local context features, or to apply weighting that decays features further away.

For aspect polarity classification, the outputs from both the GCFG and LCFG are concatenated and passed through a feature interactive learning (FIL) layer, that contains a fully connected layer and performs multi-head self-attention before classification. In contrast, aspect term extraction relies solely on the output from the GCFG to label each word in the sentence.

3
Methodology

This chapter outlines the pipeline developed for clustering and sentiment analysis of customer feedback, which can be considered to consist of the following sequential steps:

1. Preprocessing of the raw textual data
2. Generating sentence embeddings using the SBERT sentence transformer model
3. Reducing the dimensionality of the embeddings with UMAP
4. Clustering the reduced embeddings using either HDBSCAN or K-Means
5. Labeling the resulting clusters using BERTopic
6. Performing aspect-based sentiment analysis using models from the PyABSA framework

In addition, this chapter describes the experiments conducted to identify suitable hyperparameters, as well as the evaluation of the final configuration using a small human-labeled test set.

3.1 Dataset and Preprocessing

The dataset used in this project was gathered from Volvo Cars internal customer feedback platform. It is based on thousands of responses to customer surveys collected at various post-purchase intervals. Each survey asked customers how likely they were to recommend Volvo Cars to a friend or colleague, rated on an 11-point scale. This was followed by a free-text question: "*Would you mind sharing the reason for your score?*"—allowing customers to elaborate in their own words.

While the dataset includes the original free-text responses, it has also been augmented with automatic translation of comments from respective native languages into English, segmentation of longer responses into shorter sentences, and the assignment of topic labels to each sentence. While the segmentation rules were not exactly known, they were assumed to be based mainly around punctuation. For this project, the segmented sentences from the translated responses served as the primary data points and formed the basis for generating sentence embeddings, clustering, and sentiment analysis. It was assumed that most sentences reflect a single coherent theme, which is important for effective clustering.

It was decided to not make use of the numerical ratings included in the dataset for

two reasons. First, the ratings reflect the overall customer experience and are not sentence-specific. For example, a customer might be very likely to recommend Volvo Cars but write: *"The car was a pleasure to drive. But I really disliked the responsiveness of the pedals."* Although the sentences express opposing sentiments, they would share the same rating, which would be misleading for sentence-level analysis. Secondly, it would also create separation between semantically similar sentences depending on the numerical score, which is counterproductive to the primary goal of finding clusters centered around specific NVH themes regardless of sentiment.

To limit the amount of data to be processed, only survey responses collected in the last 6 months were used. Additionally, since each sentence in the dataset was accompanied by a pre-assigned topic label, these labels could be used to filter the dataset to retain only sentences related to noise, vibration, and harshness, resulting in a final subset of 36,031 sentences with an average length of 72 characters.

As a preprocessing step all sentences were lowercased and stripped of trailing special characters such as punctuation, commas and exclamation points, which was shown to have some impact on embedding positions. Furthermore, early clustering trials revealed that SBERT assigned disproportionate weight to certain words, mostly brand-specific terms such as "*volvo*", "*xc60*" or "*car*". To mitigate this, these words were all removed when generating the embeddings. This step was based on the assumption that SBERT would still be able to capture the semantic meaning of the sentences even with information missing[1].

3.2 Generating Sentence Embeddings

SBERT was chosen for generating sentence embeddings. SBERT is an extension of the original BERT architecture, specifically designed to produce semantically meaningful vector representations of text. Compared to earlier methods such as GloVe, InferSent or USE, SBERT demonstrates significantly better performance in capturing semantic and contextual information, as evident by its results on sentence similarity tasks [RG19].

This project uses the *all-MiniLM-L6-v2* model, which offers a good balance between performance and computational efficiency. It generates 384-dimensional vector embeddings, which should capture sufficient semantic information for the clustering task. It also supports a maximum input length 256 tokens, more than adequate for single-sentence inputs. Using larger SBERT variants, while increasing computational cost significantly doesn't give a corresponding gain on sentence similarity tasks to warrant their use [RG25].

Embeddings were also normalized, this was done because cosine similarity was intended to be used as the distance metric when clustering, due to its resilience to the curse of dimensionality. However, since the chosen implementations of HDBSCAN

[1]SBERT is after all based on BERT which was trained to predict missing words in sentences.

and K-Means do not support cosine similarity, Euclidean distance on normalized embeddings was used instead, which is a close approximation.

3.3 Dimensionality Reduction

After generating the sentence embeddings and conducting initial clustering trials, it became evident that HDBSCAN struggled with the high dimensionality of the original sentence embeddings. In high-dimensional spaces, due to the curse of dimensionality, all points tend to be approximately equidistant from one another. This severely hindered HDBSCAN, which relies on distinguishing between dense and sparse regions. As a result the algorithm labeled most observations as noise. To address this issue, it was decided to use a dimensionality reduction technique to improve clustering performance.

Several common dimensionality reduction techniques were considered such as Principal Component Analysis (PCA), t-distributed Stochastic Neighbor Embedding (t-SNE) and UMAP. The choice landed on UMAP because unlike PCA, which primarily captures directions of maximum global variance, and t-SNE, which emphasizes local structure, UMAP is designed to preserve both the local and global structure of the data [MHM20]. The algorithm is also highly computationally efficient, with the possibility of attaining additional speed improvements through a CUDA implementation available through the cuML library. In practice, it significantly outperforms t-SNE in terms of speed, particularly on larger datasets, making it a good choice in this context. It has also been shown to greatly improve clustering performance for HDBSCAN and K-Means across various datasets [AKC20], and is widely adopted, for example as a core component in the BERTopic topic modeling framework [Gro22].

Using UMAP for dimensionality reduction does however come at a cost: distances between points and clusters lose their meaning, and there is no longer a clear relationship between distances in the reduced space and those in the original embedding space. Points that are close in the high-dimensional space will still be close in the reduced space, but there is no guarantee that the distance between two arbitrary points is faithfully represented. As a result, any interpretation of distance or cluster geometry in the reduced space should be made with caution. It is also important to note that this means the clustering outcome is directly influenced by the UMAP parameters used.

3.4 Clustering Strategy

To group semantically similar customer feedback, clustering is performed on the reduced sentence embeddings. HDBSCAN and K-Means were selected for this task, primarily for practical reasons: the dataset is large and slow to process, and both algorithms have efficient GPU-based implementations in the cuML library, which also mirrors the well-known scikit-learn API, ensuring easy implementation and compatibility with other sklearn functions.

Early exploratory data analysis suggested that clusters would vary significantly in size and shape. This made HDBSCAN the more appropriate choice, as it is a density-based algorithm capable of identifying arbitrarily shaped clusters. In contrast, K-Means assumes clusters are spherical, equally sized, and have similar variance. Despite this, K-Means was used during the experimental phase due to its simplicity, requiring only a single parameter K to tune, its fast runtime, and because it was the only option for calculating cluster prediction strength.

3.4.1 Cluster Evaluation

Intrinsic Cluster Validation Metrics

As no ground-truth label exist for the data, a number of intrinsic cluster evaluation measures was used: Silhouette score, Davies-Bouldin Index and Calinski-Harabasz Index. These are commonly used metrics that estimate clustering quality by measuring different forms of spread and separation of clusters. Additionally the DBCV Index was included. Unlike the other metrics, DBCV is a density based metric that does not penalize non-spherical cluster shapes, making it more suitable for evaluating results from algorithms like HDBSCAN. These metrics were used exclusively to compare clusterings of the same UMAP projections, not across different reduced spaces, as suggested in Section 2.3.3.

Cluster Prediction Strength

Cluster Prediction Strength was included to address the limitations of distance-based metrics. As explained in Section 2.3.3, this method evaluates clustering stability by measuring how consistently point pairs are grouped together across clusterers trained on different subsets of the data. In this project, it was used to provide an estimate of the suitable number of clusters.

Prediction strength was computed for various values of k using K-Means for clustering algorithm. The highest value of k for which the prediction strength remains above a certain threshold was selected. The authors of [Ta05] suggest a threshold in the range of 0.8–0.9 for well-separated clusters. Given that the clusters in this dataset are not expected to be well separated, the more conservative threshold of 0.8 was used.

It is assumed that values of k showing high stability also generalize well to clustering with HDBSCAN, indicating that at least that many well-separated clusters should be recoverable using that method as well. However, since prediction strength is computed using K-Means, it may report poor stability when clusters are non-spherical or non-linearly separable, which is likely in this dataset. As a result, even if a particular value of k yields low prediction strength scores with K-Means, it could still have been possible to form k well-separated clusters using HDBSCAN. Therefore, the number of clusters suggested by prediction strength should be considered a rough estimate rather than a precise target for HDBSCAN.

Manual Inspection of Clusters

The limitations of automated evaluation metrics can also be partially offset by the fact that the data is textual, allowing for cluster comparison by manual inspection. That said, such inspection was mostly constrained by time and resources to brief glances. When inspected, clusters were judged intuitively, based on whether their content could be clearly and easily described. For instance, if a cluster consisted entirely of sentences related to the engine, or expressions of feelings about how it is to drive, it was viewed as coherent, even if containing a variety of subtopics. In contrast, if a cluster contained two logically disjoint parts, such as content about both the engine and the doors, then it was considered a poor cluster.

Cluster Labeling with BERTopic

To help manual evaluation and and speed up the interpretation of clustering results, BERTopic was used to automatically generate short descriptive cluster labels for a few select clusterings. Although BERTopic is typically used for end-to-end topic modeling [Gro22], it was here employed primarily for cluster labeling. Details of the settings and implementation are provided in Appendix A.1.

3.4.2 Experimental Setup for Hyperparameter Evaluation

Four experiments were conducted to evaluate key hyperparameters that affect the clustering of our data. The first two experiments focused on identifying UMAP settings that promote stable clustering results for K-Means, with the hope that these results also generalize to HDBSCAN. While the third and fourth experiments evaluated clustering quality for different configurations of K-Means and HDBSCAN for embeddings of different dimensionalities.

Experiment 1: Effect of Dimensionality on K-Means Cluster Stability

This experiment assessed how different embedding dimensionalities affect cluster stability and influence the number of clusters that can be reliably identified. UMAP was used for dimensionality reduction with the parameters: `n_neighbors=15`, `min_dist=0`, and the cosine distance metric. The choice of `n_neighbors=15` follows the default setting, and should emphasize local structure over global structure, which is expected to yield more separable clusters. The output dimensionality, controlled by the `n_components` parameter, was systematically varied across the values of 2, 4, 16, 32, 64, and 128. The original (non-reduced) SBERT embeddings were also included for comparison.

For each dimensionality, K-Means was applied with the `n_clusters` parameter ranging from 2 to 30 in increments of 2. Cluster stability was measured using prediction strength to quantifies how consistent cluster assignments are. This was used to estimate the optimal number of clusters. A 10-fold cross-validation split was used, as the results appeared to vary less than when using the 2 and 5-fold splits evaluated in the original cluster prediction strength paper [Ta05]. The resulting scores were averaged across splits.

Experiment 2: Effect of UMAP's Local Neighborhood Size on K-Means Cluster Stability

The second experiment aimed to evaluate how UMAP's `n_neighbors` parameter affects the number of stable clusters that can be recovered. To isolate the effect of this parameter, UMAP was configured with `n_components=64`, selected based on the results from Experiment 1, and the cosine distance metric. The `n_neighbors` parameter was varied across the values: 15, 60, 120, 240, 480, and 1000. For each setting, the resulting reduced embedding was clustered using K-Means, with `n_clusters` ranging from 2 to 30 in increments of 2. Cluster stability was again assessed using prediction strength with 10-fold cross-validation, and scores were averaged across folds.

Experiment 3: Effect of Dimensionality and `n_clusters` on K-Means Cluster Quality

This experiment aimed to evaluate how both embedding dimensionality and the number of clusters `n_clusters` affect the quality of K-Means clusterings, as measured by Silhouette score, Davies-Bouldin index, and Calinski-Harabasz score, three metrics that assess how well-separated and compact the resulting clusters are. Clustering was performed on embeddings reduced to 2, 4, 16, 32, 64, and 128 dimensions, as well as on the original non-reduced 384-dimensional SBERT embeddings. For each dimensionality, K-Means was applied with `n_clusters` ranging from 2 to 30 in increments of 2.

Experiment 4: Effect of Dimensionality and `min_cluster_size` on HDBSCAN Cluster Quality

This experiment aimed to evaluate the effect of the `min_cluster_size` parameter in HDBSCAN for different embedding dimensionalities. As in previous experiments, clustering was performed on embeddings reduced to 2, 4, 16, 32, 64, and 128 dimensions, as well as the original non-reduced 384-dimensional SBERT embeddings. Dimensionality reduction was carried out using UMAP with `n_neighbors=15` and cosine distance. For each dimensionality, `min_cluster_size` was varied from 100 to 1000 in increments of 100. HDBSCAN was configured with `min_samples=7`, as larger values tended to produce more noise points than desirable. All other parameters were kept at their default values.

Cluster quality in this experiment was measured using the DBCV score, which evaluates the density and separation of clusters. In addition to DBCV, the number of points classified as noise and the total number of clusters formed were recorded. These supplementary metrics provide important context for evaluating the HDBSCAN results, as DBCV score alone does not reflect the proportion of noise points or how many clusters were found.

3.4.3 Selecting the Final Clustering Configuration

The selection of the configuration for the clustering part of the pipeline was guided by the results of the previous experiments, but also relied on intuition and input from domain experts. While metrics like DBCV, silhouette score, and others provided helpful guidance, they do not fully reflect semantic coherence, topic relevance, or interpretability, and was therefore not used as the sole basis for model selection.

Evaluating The Final Clustering Configuration

As a final step in evaluating whether the clustering model produces clusters that represent coherent topics that easily can be understood by a human. A small classification task was designed to measure the degree of agreement between the model and a human annotator when labeling new observations. A high level of agreement would suggest that the cluster labels are meaningful and that the underlying clusters are semantically coherent.

For this purpose a test set of 200 observations were sampled from a separate but related dataset of customer feedback. Unlike the primary dataset described in Section 3.1, which is survey-based, this dataset consists of voluntary feedback submitted by customers via a mobile app provided by Volvo Cars. This data tends to be more negatively skewed, as users are more likely to leave feedback when encountering issues. The distribution of topics also differs, both due to the prevalence of more negative feedback and the introduction of additional topics, since this dataset could not be filtered to only contain NVH-related content.

Each test observation was manually assigned a cluster label corresponding to one of the clusters produced by the selected HDBSCAN model. These manual labels served as ground truth, against which the model's predicted cluster assignments were compared. To obtain model predictions, each test observation had to first be transformed using the previously trained UMAP model, ensuring they were embedded in the same space as the original data. The transformed embeddings were then assigned cluster labels using HDBSCAN's internal `approximate_predict` method.

Two annotators were involved in the manual labeling process, each following a different strategy. One annotator assigned labels based on semantic similarity—choosing the cluster whose label best matched the meaning of the observation. The other adopted a root-cause strategy: for example, if an observation mentioned a display issue but the underlying cause was known to be software-related, the label would be assigned to software if such a label existed, or otherwise as noise, rather than display.

Given that this is a multi-class problem and large class imbalances are expected, weighted F1-score was chosen as the primary evaluation metric, as it accounts for both precision and recall while adjusting for class frequencies. This provides a better assessment than simple accuracy in a class imbalanced setting.

3.5 Sentiment Extraction

For sentiment analysis, the PyABSA framework [YZL23] was used, which offers a collection of pre-trained models for aspect-based sentiment analysis. The selected ABSA model was the LCF-ATEPC model, introduced in [YZY+21], which ranks among the top five models on the SemEval-2014 Task 4 benchmark [PGP+14], a benchmark evaluating ABSA performance on customer reviews from the restaurant and laptop domains.

For sentence-level sentiment analysis, the DeBERTa-v3-base-absa-v1.1 model was used, also available through the PyABSA framework and hosted on Hugging Face at `https://huggingface.co/yangheng/deberta-v3-base-absa-v1.1`. This model was selected for its ease of implementation.

Both models were applied independently to each sentence in the dataset to generate aspect-based and sentence-level sentiment predictions.

4
Results and Analysis

This chapter presents the results from the final clustering and sentiment analysis pipeline, covering key design choices, hyperparameter settings, the resulting clusters, as well as evaluation results on a human-labeled test set. This is then followed by a comparison of K-Means and HDBSCAN in terms of clustering quality at different levels of clustering granularity. The chapter concludes with results from the hyperparameter tuning experiments for UMAP and HDBSCAN. Results from K-Means hyperparameter experiments are provided in Appendix A.2. For confidentiality, all cluster labels and extracted aspects shown in the figures have been anonymized.

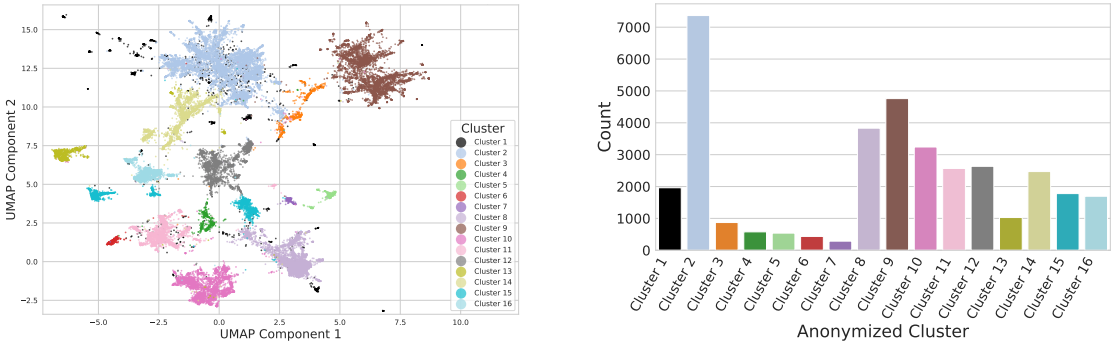
4.1 Final Clustering Model

4.1.1 Parameters and Model Choice

Based on the comparisons in Section 4.2, HDBSCAN was selected as clustering method due to its consistent ability to generate semantically coherent and distinct clusters at different levels of clustering granularity. Although the experimental results (see Section 4.3) suggested 64-dimensional embeddings as the best choice, further testing revealed that the 32-dimensional configuration yielded clusters that were more intuitive and better aligned with domain-relevant categories. As a result, the 32-dimensional embeddings were chosen for the final model.

The final model was configured with `min_samples = 7` and `min_cluster_size = 200`. This setup initially produced more clusters than desired, thus the model was further refined by tuning `cluster_selection_epsilon` to a value of 0.26. This was done to strike a balance between avoiding overly specific clusters and preserving important topic distinctions. Priority was given to avoiding the merging of semantically distinct clusters. To counteract the resulting fragmentation of clusters, the final step involved manually merging some clusters where it was deemed appropriate. Some automatically generated cluster labels were also adjusted to better reflect the underlying content. This ensured that the final set of categories remained both manageable and relevant to the needs of the teams at the NVH department. In the end, this process resulted in 15 coherent clusters, not counting Cluster 1 which contains the data points that HDBSCAN identify as noise[1]. These clusters span a wide range of themes, from general ownership experience to more specific feedback regarding individual car components (see Figure 4.1).

[1]Cluster 1 will represent the points identified as noise in all HDBSCAN visualizations.



(a) A two-dimensional UMAP projection of clustered sentences, each point representing a sentence and colors indicating cluster membership.

(b) Distribution of observations per cluster, showing variation in cluster sizes.

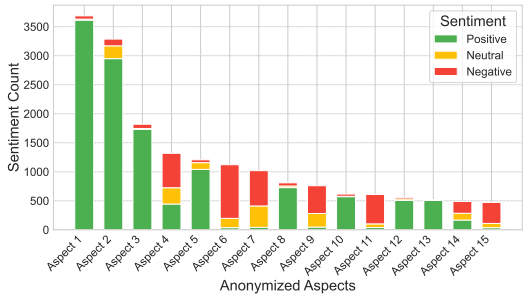
Figure 4.1: Clustering results for the final configuration of the clustering pipeline.

4.1.2 Results on Test-set

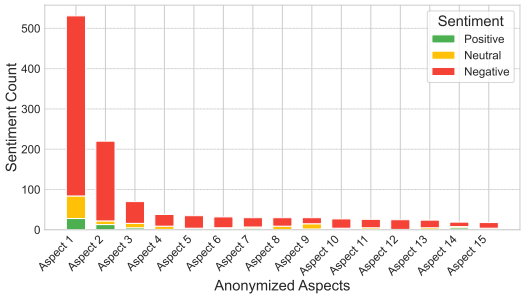
To evaluate the interpretability and practical usefulness of the final clustering model, a test set consisting of customer feedback sentences from a different data source was annotated by human labelers (see Section 3.4.3). This evaluation did not aim to determine whether the clustering itself was objectively "correct". Rather it was to determine how well the clusters can be interpreted and described by a human, which serves as an indirect measure of their quality and coherence given that you accept the clustering as correct.

Two evaluation scenarios were conducted. In the first, the annotator labeled each sentence based on its semantic similarity to the model's cluster labels. Under this criterion, the model achieved an accuracy of 0.599 and a weighted F1 score of 0.561. This indicates that the annotator's judgments are somewhat consistent with the model's predictions.

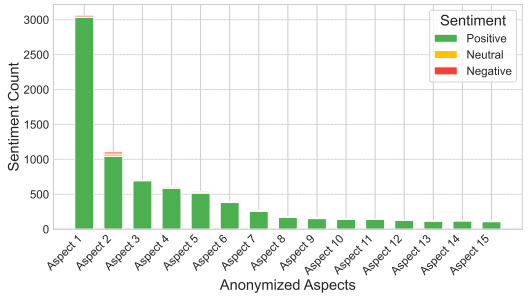
A closer inspection revealed that most misclassified sentences were concentrated around two specific clusters. In the first case, many of the sentences plausibly belonged to two different topics, making the task of assigning a single label both difficult and highly subjective. This also conflicted with one of our key assumptions, that each sentence belongs to a single clear topic. The second case involved sentences whose true labels were noise, but which the model had misclassified into various other clusters. This, is most likely due to a mismatch between the test and training data distributions. The test data could not be filtered in the same way as the training data and therefore included previously unseen topics, such as software-related comments. This may have contributed to incorrect cluster assignment, as UMAP has known limitations in handling out-of-distribution data and struggles to adequately represent novel "classes" not seen during training in the reduced embedding space [SMG20].



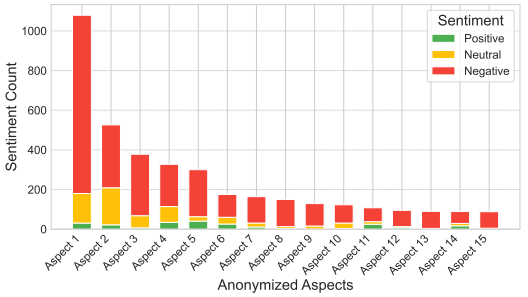
(a) Aspect sentiment distribution for the entire dataset.



(b) Aspect sentiment distribution for a component specific cluster.



(c) Aspect sentiment distribution for a cluster centered around general ownership experience.



(d) Aspect sentiment distribution for another component specific cluster.

Figure 4.2: Distribution of aspect-based sentiments for the most frequent extracted aspects for selected clusters. Bar height indicates aspect frequency; color denotes sentiment polarity (positive, neutral, or negative).

Despite these limitations, the model performed well in several categories, particularly those that were less ambiguous, with some achieving 100% classification accuracy. However, due to the small and imbalanced nature of the test set, several clusters were underrepresented or entirely absent. Consequently, performance estimates for these clusters are limited in reliability and should be interpreted with caution.

In the second scenario, the annotator performed a light root cause analysis, labeling each sentence based on the underlying issue rather than purely on semantic similarity. Here, performance dropped down to roughly 0.341 for Accuracy and 0.286 for weighted F1 score, highlighting a core limitation of the current approach: clustering based solely on sentence-level embeddings is insufficient to reflect underlying root causes or the reasoning of a domain expert.

4.1.3 Sentiment Trends

The extracted aspect-level sentiments offer additional insight into the composition of each cluster. Although the aspect terms have been anonymized for confidentiality, it can be said that their distributions generally align with the themes implied by the

cluster labels. Some clusters are predominantly negative (Figure 4.2b), while others are more positive (Figure 4.2c). This supports earlier K-Means results suggesting a split between general, positively framed experiences and component-specific complaints.

On a broader level, the full dataset (Figure 4.2a) exhibits an aspect sentiment distribution that tilts toward the positive. This likely reflects the fact that, although most clusters are predominantly negative, the largest cluster by volume is characterized by generally positive sentiment, which has a dominant influence on the overall sentiment polarity distribution.

It is important to note however, that the accuracy of the extracted aspects and sentiment polarities has not been evaluated. Therefore, while they can serve as supporting evidence for the validity of the cluster assignments, one should avoid drawing overly confident conclusions about the specific aspects or sentiments themselves.

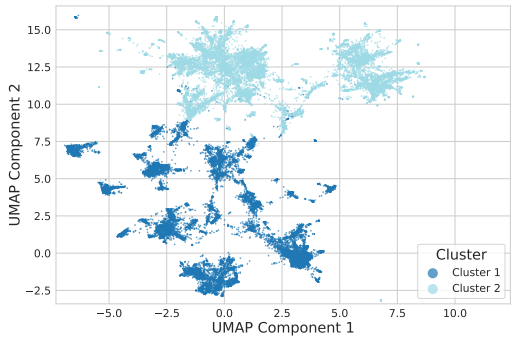
4.2 Comparing K-Means and HDBSCAN Clusterings Qualitatively

To assess the quality of the clustering approaches explored in this thesis, intrinsic evaluation metrics alone are insufficient. While useful, they do not always reflect how interpretable the resulting clusters are from a human perspective. Therefore, a qualitative comparison was conducted to evaluate how effectively each method grouped semantically related feedback at varying levels of clustering granularity.

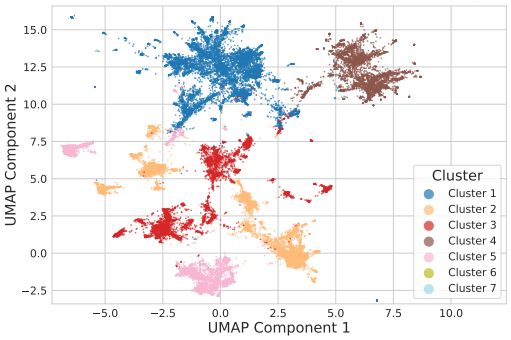
For K-Means, clusterings with $k = 2$, $k = 7$, and $k = 14$ were selected based on strong performance across three intrinsic metrics (see Figure 4.4). For HDBSCAN, `min_cluster_size` values of 1000, 700, and 500 were chosen based on the resulting number of clusters, proportion of noise points, and DBCV score (Figure 4.5), to yield different levels of clustering granularity. The `min_samples` parameter were fixed at 7 and all other parameters left at their default settings. All clusterings were performed on the same 64-dimensional reduced embeddings. Note that the 2D scatterplots in Figure 4.3 serve only as visual approximations and do not exactly represent the structure of the data in the space it was clustered in.

At $k = 2$, the clusters are broad but meaningful: one cluster capturing largely positive general impressions about driving and ownership experience. The other cluster (Cluster 2 in Figure 4.3a) captures a wide range of feedback related to different car components, reflected in the visualization by its dispersed structure across several isolated groups. Despite this fragmentation, the points can still be reasonably interpreted as belonging to a common theme of component-related feedback.

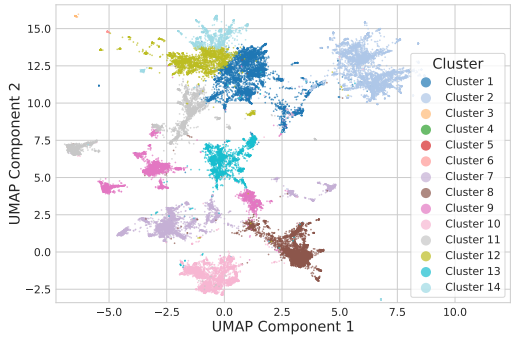
At $k = 7$, coherence of clusters weaken. Some clusters remain semantically consistent, while others now consist of disparate themes that cannot be efficiently summarized. For example, Cluster 2 and Cluster 3 (Figure 4.3b) contain feedback related



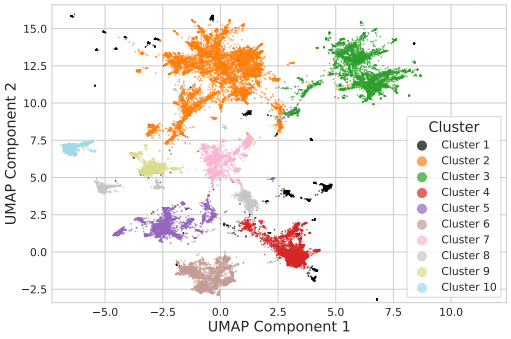
(a) K-Means, $k = 2$



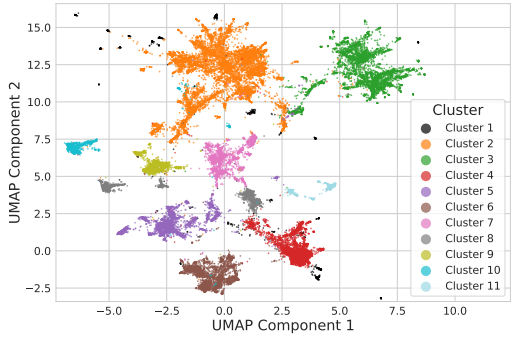
(b) K-Means, $k = 7$



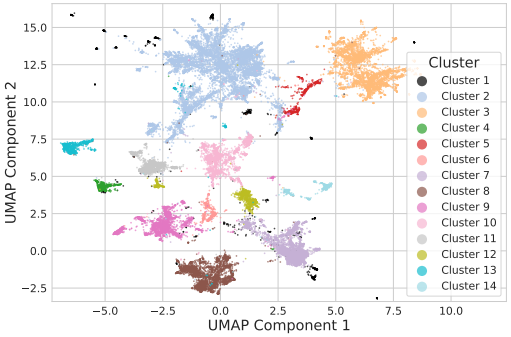
(c) K-Means, $k = 14$



(d) HDBSCAN, `min_cluster_size`=1000



(e) HDBSCAN, `min_cluster_size`=700



(f) HDBSCAN, `min_cluster_size`=500

Figure 4.3: Two-dimensional Visualizations of clusterings produced by K-Means and HDBSCAN reflecting different levels of clustering: from broad groupings, to intermediate clusters, to more fine-grained partitions.

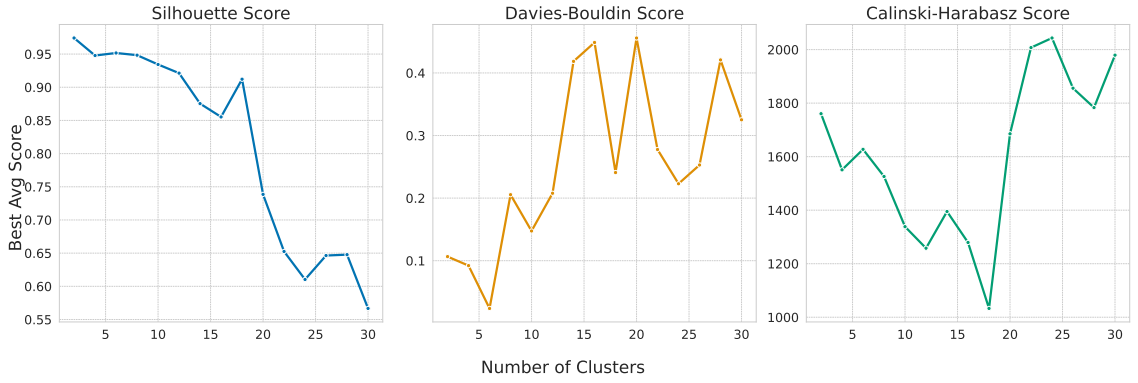


Figure 4.4: K-Means clustering metrics for different values of k on 64-dimensional embeddings. Scores are averaged over 10 runs.

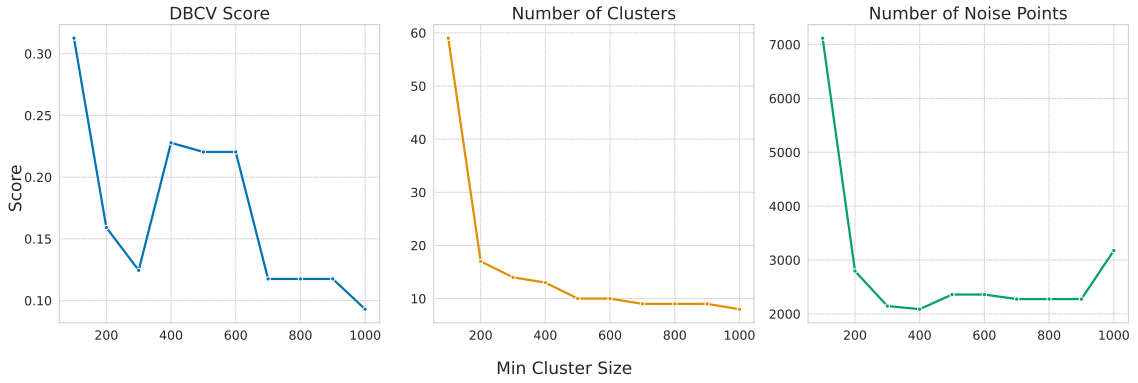


Figure 4.5: HDBSCAN clustering metrics for different values of `min_cluster_size` on 64-dimensional embeddings. Scores are from a single run as they did not vary.

to various car components that a human would not intuitively group together, making them difficult to label meaningfully. There is also an extremely small cluster (Cluster 6), which appears almost negligible. This highlights a key limitation of K-Means: it will always partition the data into exactly k clusters, even when fewer would have sufficed.

At $k = 14$, the clustering quality deteriorates further. Many clusters are either too small to be useful or appear redundant, covering similar themes with minimal distinction, often splitting previously coherent clusters in ways that reduce interpretability. For example, Clusters 1, 12, and 14 (Figure 4.3c) all contain nearly identical feedback related to driving experience. These previously belonged to the same cluster and should ideally have remained as one, but have now been fragmented into multiple clusters with nearly identical content.

In contrast, HDBSCAN proves more robust across all levels of clustering granularity (Figure 4.3d-f). Even when forming 13 clusters (excluding the noise points), the resulting groups are intuitive, meaningful, and non-overlapping. This supports the expectation that HDBSCAN is better suited to capturing the underlying structure

of the data, clearly reflected in how well it separates the various "islands" in the figures. Unlike K-Means, HDBSCAN does not require specifying the number of clusters in advance and tends to avoid producing very small or fragmented clusters. However, setting the `min_cluster_size` parameter too high can cause smaller but meaningful groups to be labeled as noise.

However, some clusters showed ambiguity in how they should be interpreted or separated. Specifically, types of feedback that could reasonably be viewed as distinct were sometimes grouped together. For example, comments directly about the engine were often merged with feedback related to driving experience. Whether this is appropriate will of course depend. Since engine performance directly affects driving experience, combining them is logically justifiable. If the goal is to maintain a clear separation between component-specific and experience-based feedback (as seen in the $k = 2$ K-Means clustering), or if isolating engine related issues is a priority, then these topics should ideally be separated. Achieving this may require further parameter tuning, such as adjusting `min_cluster_size` in conjunction with `cluster_selection_epsilon`.

Overall, HDBSCAN outperformed K-Means by consistently producing semantically coherent and practically useful clusters across different levels of clustering granularity. While K-Means showed reasonable performance at $k = 2$, its ability to form distinct and interpretable clusters declined as k increased. These results demonstrate the superiority of HDBSCAN over K-Means in capturing the underlying structure of the data, and underscore the importance of looking beyond intrinsic metrics when evaluating clustering quality.

4.3 Results from Hyperparameter Evaluation Experiments

This section summarizes the results from three of the four hyperparameter evaluation experiments that guided the development of the final clustering model. Results for the fourth experiment, evaluating K-Means clustering quality across different dimensionalities, are included in Appendix A.2.

Experiment 1: Effect of Dimensionality on K-Means Cluster Stability

Figure 4.6 shows the average prediction strength for K-Means clustering across different values of k and UMAP output dimensionalities. The goal was to assess how the number of output dimensions (`n_components`) affects the stability of clustering results. Embeddings with 64 and 128 dimensions produced significantly more stable clusters compared to all other dimensionalities. By comparison, lower-dimensional embeddings as well as the original embeddings, showed markedly worse stability even for small values of k and got progressively worse as the number of clusters increased. These results suggest that for K-Means to reliably identify and separate distinct clusters, the dimensionality of the embedding needs to be sufficiently high.

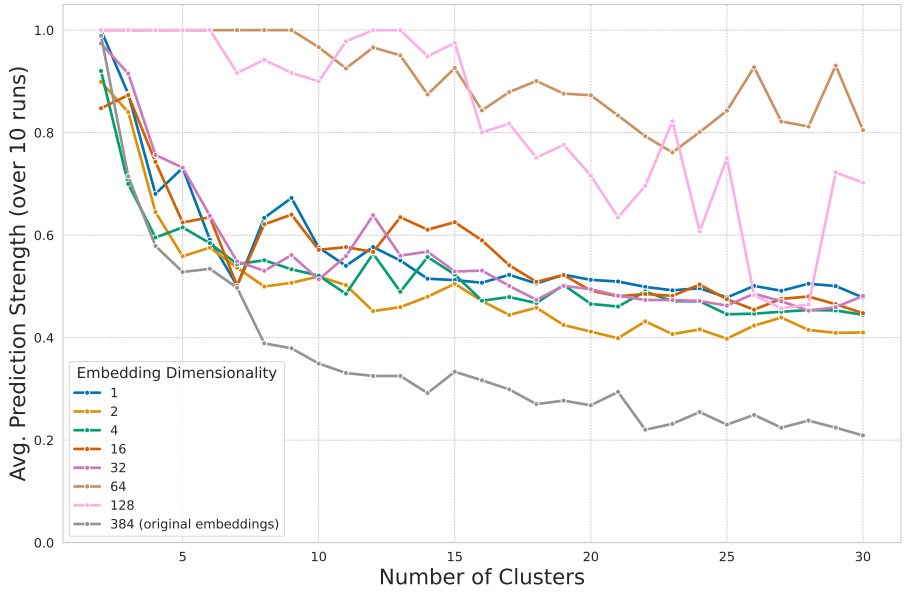


Figure 4.6: Average prediction strength using K-Means across different values of k and UMAP output dimensionalities (`n_components`).

A possible explanation is that lower-dimensional embeddings yield less clearly defined clusters with more overlapping boundaries, making them harder for K-Means to separate. Based on these results, 64 dimensions were considered a promising setting. This was preferred over 128 dimensions, both because it performed better, but also because HDBSCAN performance tends to degrade sharply with dimensionalities approaching 100, making lower-dimensional embeddings preferable.

Experiment 2: Effect of UMAP's Local Neighborhood Size on K-Means Cluster Stability

Figure 4.7 illustrates how varying the `n_neighbors` parameter in UMAP affects cluster stability, as measured by average prediction strength across different values of k. In this experiment, the embedding dimensionality was fixed at 64 based on the findings from Experiment 1, while the neighborhood size was varied. The results show that a smaller neighborhood size of 15 produced the highest number of stable clusters, with stability gradually declining as `n_neighbors` increase. This suggests that emphasizing local structure over global structure in the UMAP projection leads to embeddings where the clusters are more clearly-separated. Based on this, `n_neighbors=15` was selected for subsequent experiments.

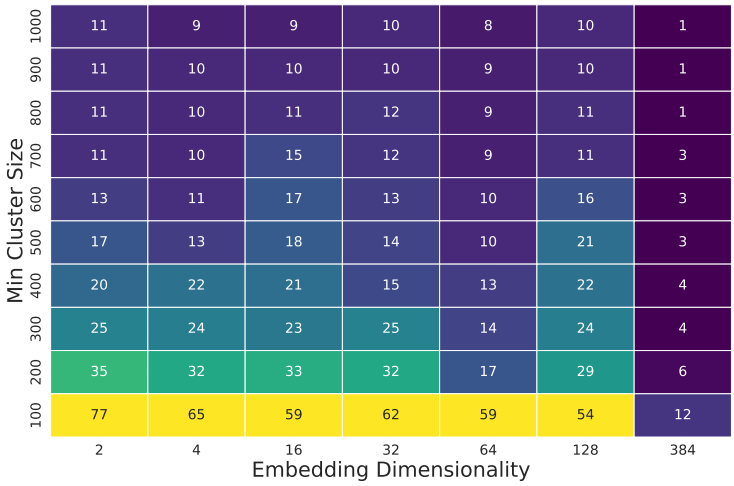
Experiment 4: Effect of Dimensionality and `min_cluster_size` on HDBSCAN Cluster Quality

Figure 4.8 presents three heatmaps showing HDBSCAN performance across a range of different dimensionalities and `min_cluster_size` values. These results highlight the trade-off between identifying compact, high-density clusters and limiting the number of points classified as noise. Higher DBCV scores are generally associated

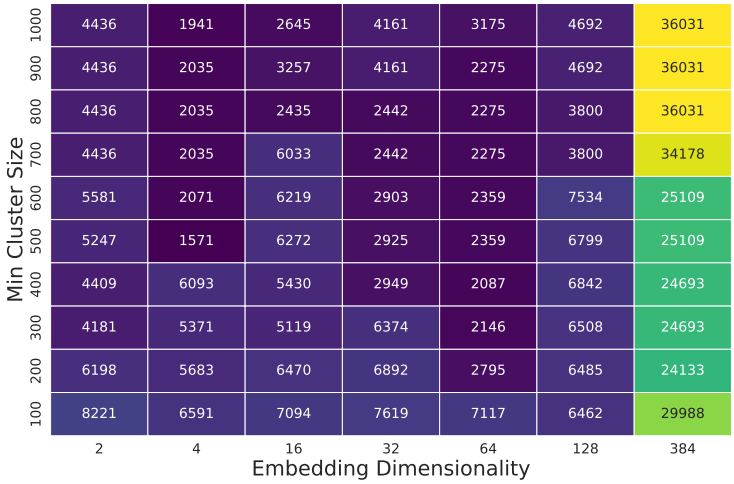
Figure 4.7: Average prediction strength using K-Means across different values of k and UMAP `n_neighbors` settings.

with configurations that also produce a higher proportion of noise. They also show that the original 384-dimensional embeddings performed extremely poor, with most data points classified as noise. On the other hand, dimensionalities of 32 to 64 combined with `min_cluster_size` values of 400–600 strike a good balance between cluster quality and proportion of points considered noise. This configuration also yields a moderate number of clusters, supporting the practical goal of finding meaningful but not overly fragmented topics.

(a) DBCV Scores



(b) Number of clusters



(c) Number of noise points

Figure 4.8: Clustering performance of HDBSCAN across combinations of embedding dimensionality and `min_cluster_size`. Subfigures show DBCV scores, number of clusters, and number of noise points, respectively.

5
Conclusions

This chapter addresses the research questions, discusses the limitations of the chosen approach, and outlines possible directions for future work.

5.1 Addressing the Research Questions

Can consumer feedback be clustered in such a way that each cluster aligns with specific NVH-related topics handled by different teams within the NVH department at Volvo? If not, do other meaningful groupings emerge that still provide value?

It was not possible to cluster customer feedback in a way that directly matched the internal structure of the Vehicle Attributes department, but the final configuration of the pipeline still uncovered useful groupings. It produced 15 interpretable clusters (excluding the points identified as noise by HDBSCAN), capturing semantically coherent and logically related themes ranging from technical issues and specific components to broader impressions of vehicle ownership and the driving experience. These clusters remain valuable, as they organize feedback into coherent groupings around recurring themes, helping teams more efficiently identify information relevant to their work.

The interpretability of the discovered clusters was supported by a classification-based evaluation, in which predicted cluster labels were compared to those a human annotator would have assigned by selecting the most semantically appropriate cluster label for each sentence. The resulting weighted F1 score of approximately 0.56 suggests that the semantic themes captured were generally meaningful and recognizable to a human. Several factors may however have negatively impacted this score, including the limited size and representativeness of the test set, UMAP's inability to produce appropriate reduced embeddings for out-of-distribution inputs, and the challenge of assigning a single topic label to feedback that may span multiple semantic categories.

Can sentiment analysis applied to these clusters yield insights that help in understanding general customer opinions about specific aspects of the vehicle?

Yes, sentence-based sentiment analysis applied at the cluster level proved effective

in providing insights into general customer opinions about various aspects of the vehicle. Even without extracting individual aspects, sentiment polarity at the cluster level was often sufficient to support interpretation of how different components or experiences were perceived, though this depended in large part on the granularity of the final clustering.

Aspect-based sentiment analysis was also applied to uncover more specific subtopics within each cluster. While the overall distribution of extracted aspects generally matched the theme of their respective clusters, closer examination revealed that individual aspect terms were often inaccurate, inconsistently extracted or added limited additional value. As such, ABSA served better as an indicator of cluster validity than as a source of further information.

5.2 Limitations and Future Work

A core limitation of this approach lies in the reliance on semantic similarity as the basis for clustering, which cannot account for underlying technical root causes behind what the customer is describing. For example, similar-sounding feedback may stem from entirely different vehicle systems and end up in the same cluster. While a single technical issue may be described in different ways and end up scattered across multiple clusters. This complicates efforts for teams to find and extract feedback relevant to their specific areas of responsibility. Despite this, the groupings provide a useful starting point for manual review, especially when interpreted by individuals with relevant domain knowledge.

Another limitation is generalizability. The pipeline may perform poorly on new data that differ too much from the training data. HDBSCAN cannot incorporate new clusters after training, and UMAP performs poorly on out-of-distribution data [MHS24], often embedding unfamiliar inputs into existing clusters. A potential solution to the latter issue is to adopt Parametric UMAP [SMG20], which supports retraining to adjust the mapping between the original embedding space and the reduced space. This allows the projection to adapt to shifts in the data distribution, increasing the likelihood that unfamiliar inputs are separated from existing clusters. Another solution is to, when possible filter the data beforehand.

Evaluation of the pipeline was also limited. Only 200 manually labeled samples were used to assess 15 clusters, plus the points labeled as noise by HDBSCAN as an additional category. As a result, several clusters were underrepresented or entirely absent in the test set. Additionally, the sentiment analysis results were not formally validated. To ensure reliable conclusions, a more thorough evaluation is needed. This includes using test data that reflects the distribution of data the pipeline is intended to process, labeling enough samples to cover all clusters, and formally verifying the consistency and accuracy of the sentiment analysis models.

Additionally, while a model with a low noise ratio was chosen to retain as much potentially valuable input as possible, included noise may distort sentiment distri-

butions and make it harder to find relevant information. A reliable estimate of the expected noise proportion could have guided model selection toward more effective noise filtering without sacrificing meaningful content.

Finally, the sentiment analysis component could benefit from domain adaptation as it is strongly suggested to fine-tune models on domain-specific datasets for better performance. Using more advanced techniques, such as Aspect Sentiment Triplet Extraction (ASTE), which extracts aspects, opinions, and their sentiments, instead of ATEPC could also make the ABSA component more informative. One promising such model is Bidirectional Machine Reading Comprehension (BMRC) [CWLW21], which has demonstrated strong performance on Car-MuSe, a dataset specifically developed for ASTE tasks in the automotive domain [UHJ+25]. Although Car-MuSe is restricted to non-commercial use, a similar in-house dataset could likely be developed to support either fine-tuning of the current ATEPC model or the adoption of ASTE models like BMRC.

Bibliography

[AA15] Fors C Genell A Anund A, Lahti E. The effect of low-frequency road noise on driver sleepiness and performance. *PLoS ONE*, 10(4):e0123835, 2015.

[AKC20] Mebarka Allaoui, Mohammed Lamine Kherfi, and Abdelhakim Cheriet. Considerably improving clustering algorithms using umap dimensionality reduction technique: A comparative study. In Abderrahim El Moataz, Driss Mammass, Alamin Mansouri, and Fathallah Nouboud, editors, *Image and Signal Processing*, pages 317–325, Cham, 2020. Springer International Publishing.

[Ala19] Jay Alammar. The illustrated bert, elmo, and co., 2019.

[ATOS23] Majid Hameed Ahmed, Sabrina Tiun, Nazlia Omar, and Nor Samsiah Sani. Short text clustering algorithms, application and challenges: A survey. *Applied Sciences*, 13(1), 2023.

[BAPM15] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference, 2015.

[CDA+17] Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In Steven Bethard, Marine Carpuat, Marianna Apidianaki, Saif M. Mohammad, Daniel Cer, and David Jurgens, editors, *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[CKS+18] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data, 2018.

[CMS13] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. Density-based clustering based on hierarchical density estimates. In Jian Pei, Vincent S. Tseng, Longbing Cao, Hiroshi Motoda, and Guandong Xu, editors, *Advances in Knowledge Discovery and Data Mining*, pages 160–172, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[CWLW21] Shaowei Chen, Yu Wang, Jie Liu, and Yuelin Wang. Bidirectional machine reading comprehension for aspect sentiment triplet extraction. *CoRR*, abs/2103.07665, 2021.

[CYyK+18] Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil.

Universal sentence encoder, 2018. Additional information available at https://tfhub.dev/google/universal-sentence-encoder/4.

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[Des16] Bernard Desgraupes. Clustering indices. 2016.

[GGKC21] Benyamin Ghojogh, Ali Ghodsi, Fakhri Karray, and Mark Crowley. Uniform manifold approximation and projection (umap) and its variants: Tutorial and survey, 2021.

[Gro22] Maarten Grootendorst. Bertopic: Neural topic modeling with a class-based tf-idf procedure, 2022.

[IEA+23] Abiodun M. Ikotun, Absalom E. Ezugwu, Laith Abualigah, Belal Abuhaija, and Jia Heming. K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. *Information Sciences*, 622:178–210, 2023.

[JMN23] S. Johnson, M. Murty, and I. Navakanth. A detailed review on word embedding techniques with emphasis on word2vec. *Multimedia Tools and Applications*, 83:1–29, 10 2023.

[McI25] Healy J. Melville J. McInnes, L. Umap documentation: How umap works. https://umap-learn.readthedocs.io/en/latest/how_umap_works.html, 2025.

[MHA25a] Leland McInnes, John Healy, and Steve Astels. Hdbscan documentation: Faq. https://hdbscan.readthedocs.io/en/latest/faq.html, 2025.

[MHA25b] Leland McInnes, John Healy, and Steve Astels. Hdbscan documentation: How hdbscan works. https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html, 2025.

[MHA25c] Leland McInnes, John Healy, and Steve Astels. Hdbscan documentation: Parameter selection. https://hdbscan.readthedocs.io/en/latest/parameter_selection.html„ 2025.

[MHM20] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020.

[MHS24] Leland McInnes, John Healy, and Nathaniel Saul. Umap documentation: Transforming new data with parametric umap. https://umap-learn.readthedocs.io/en/latest/transform_landmarked_pumap.html, 2024.

[MJC+] Davoud Moulavi, Pablo A. Jaskowiak, Ricardo J. G. B. Campello, Arthur Zimek, and Jörg Sander. *Density-Based Clustering Validation*, pages 839–847.

[PGP+14] Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. SemEval-2014 task 4: Aspect based sentiment analysis. In Preslav Nakov and Torsten Zesch, editors, *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35, Dublin, Ireland, August 2014. Association for Computational Linguistics.

[PGW25] Dehua Peng, Zhipeng Gui, and Huayi Wu. Interpreting the curse of dimensionality from distance concentration and manifold effect, 2025.

[QAPS09] Mohamad S. Qatu, Mohamed Khalid Abdelhamid, Jian Pang, and Gang Sheng. Overview of automotive noise and vibration. *International Journal of Vehicle Noise and Vibration*, 5:1–35, 2009.

[RG19] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

[RG25] Nils Reimers and Iryna Gurevych. Sentence-transformers documentation, 2025.

[SMG20] Tim Sainburg, Leland McInnes, and Timothy Q. Gentner. Parametric UMAP: learning embeddings with deep neural networks for representation and semi-supervised learning. *CoRR*, abs/2009.12981, 2020.

[Ta05] Robert Tibshirani and Guenther Walther and. Cluster validation by prediction strength. *Journal of Computational and Graphical Statistics*, 14(3):511–528, 2005.

[TR16] Nenad Tomašev and Miloš Radovanović. *Clustering Evaluation in High-Dimensional Data*, pages 71–107. Springer International Publishing, Cham, 2016.

[UHJ+25] Atiya Usmani, Saeed Hamood Alsamhi, Muhammad Jaleed Khan, John Breslin, and Edward Curry. Muse-caraste: A comprehensive dataset for aspect sentiment triplet extraction in automotive review videos. *Expert Systems with Applications*, 262:125695, 2025.

[WKR24] Mayur Wankhade, Chaitanya Kulkarni, and Annavarapu Chandra Sekhara Rao. A survey on aspect based sentiment analysis methods and challenges. *Applied Soft Computing*, 167:112249, 2024.

[WM22] Kulkarni Chaitanya Wankhade Mayur, Rao Annavarapu Chandra Sekhara. A survey on sentiment analysis methods, applications, and challenges. *Artificial Intelligence Review*, 55(7), 2022.

[YZL23] Heng Yang, Chen Zhang, and Ke Li. Pyabsa: A modularized framework for reproducible aspect-based sentiment analysis. In Ingo Frommholz, Frank Hopfgartner, Mark Lee, Michael Oakes, Mounia Lalmas, Min Zhang, and Rodrygo L. T. Santos, editors, *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, CIKM 2023, Birmingham, United Kingdom, October 21-25, 2023*, pages 5117–5122. ACM, 2023.

[YZY+21] Heng Yang, Biqing Zeng, Jianhao Yang, Youwei Song, and Ruyang Xu. A multi-task learning model for chinese-oriented aspect polarity classification and aspect term extraction. *Neurocomputing*, 419:344–356, 2021.

[ZLD+22] Wenxuan Zhang, Xin Li, Yang Deng, Lidong Bing, and Wai Lam. A survey on aspect-based sentiment analysis: Tasks, methods, and challenges, 2022.

[ZPS+24] Charles Zhang, Benji Peng, Xintian Sun, Qian Niu, Junyu Liu, Keyu Chen, Ming Li, Pohsun Feng, Ziqian Bi, Ming Liu, Yichao Zhang, Cheng Fei, Caitlyn Heqi Yin, Lawrence KQ Yan, and Tianyang Wang.

From word vectors to multimodal embeddings: Techniques, applications, and future directions for large language models, 2024.

A

Appendix 1

A.1 BERTopic Configuration Settings

This appendix describes the configuration used when applying BERTopic for cluster labeling, as described in Section 3.4.1.

Embeddings and Clustering
To ensure consistency across everything I did, precomputed embeddings that had already been reduced with UMAP were reused as input to the BERTopic model. This approach allowed for consistent representation of the data both within BERTopic and when doing experiments or data exploration outside of BERTopic. Clustering was performed using either HDBSCAN or K-Means, with parameter settings that had been identified as promising in earlier experiments, see Section 3.4.2.

Vectorization and Topic Representation

- **Vectorizer:** `CountVectorizer` with the following configuration:
  - `ngram_range=(1, 2)`
  - `max_features=15000`
  - `stop_words=custom` - In addition to standard English stopwords, the terms `"car"`, `"cars"`, `"Car"`, and `"Cars"` were excluded, as they were very frequent and uninformative as all clusters in the dataset inherently relate to cars.
- **Representation Models:** Two representation models were chained together to fine-tune the topic representations and generate the cluster labels.
  1. **Maximal Marginal Relevance (MMR):** Applied with a diversity parameter of 0.5 to select a representative and diverse subset of documents from each cluster.
  2. **LLM Labeling:** Cluster summaries were generated using the `Zephyr-7B` language model. Fifteen documents from each cluster were passed to the model to produce a label.

Prompt Used for Label Generation
The following prompt was provided to the language model to generate the labels:

```
prompt = """<|system|>You are a helpful, respectful and honest assistant for
    labeling topics..</s>
<|user|>
I have a topic that contains the following documents: [DOCUMENTS]
The topic is described by the following keywords: '[KEYWORDS]'.
```

```
Based on the information about the topic above, please create a short label that
    captures which area, aspect, or component of the car the topic regard.
Try to make it general.
Make sure you only return the label and nothing more.</s>
<|assistant|>"""
```

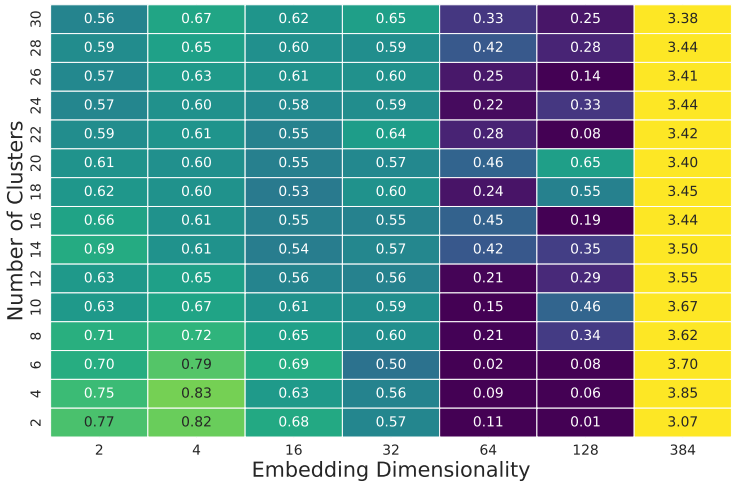
Listing A.1: Prompt used for cluster label generation

A.2 The Results for Experiment 3: Effect of Dimensionality and k on K-Means Cluster Quality

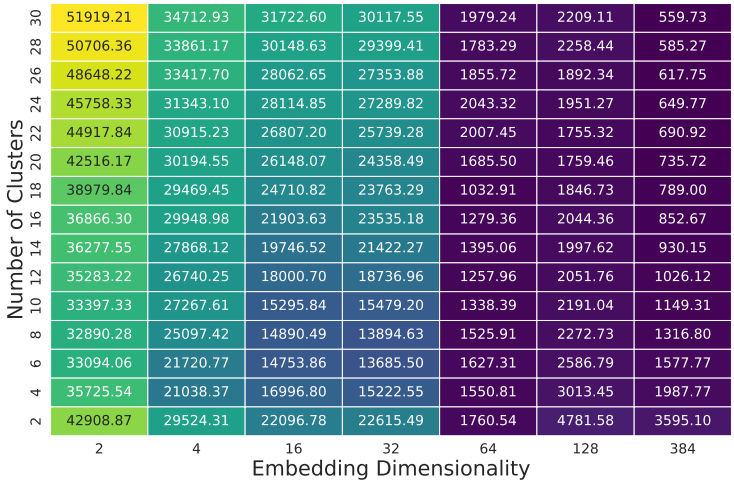
Figure A.1 presents heatmaps illustrating how K-Means clustering performance varies with embedding dimensionality and the number of clusters k. Across all three metrics, the original 384-dimensional embeddings perform the worst, which is expected. Applying UMAP shifts points closer together and tends to form tighter, more well-defined clusters, regardless of the reduced dimensionality. A notable shift in scores occurs when increasing the dimensionality from 32 to 64: Silhouette and Davies-Bouldin scores improve markedly, while the Calinski-Harabasz score drops. This divergence likely reflects a known bias in the former two metrics toward higher-dimensional spaces, as noted in [TR16]. Regardless, intrinsic clustering scores should not be compared across different UMAP dimensionalities, as the transformation alters the data structure in ways that distort metric values, irrespective of actual clustering quality.

Regarding the optimal number of clusters, Silhouette score strongly favors $k = 2$ across most dimensionalities, except the lowest. In contrast, Davies-Bouldin and Calinski-Harabasz scores often indicate much higher optimal values, typically ranging from 18 to 30 at lower dimensions and dropping to 2 at highest ones. These inconsistencies highlight the problem with relying solely on intrinsic metrics to determine either the optimal number of clusters or the ideal embedding dimensionality.

(a) Silhouette Scores



(b) Davies-Bouldin Scores



(c) Calinski-Harabasz Scores

Figure A.1: Clustering metrics for K-Means across combinations of embedding dimensionality and number of clusters k. Subfigures show the Silhouette score, Davies-Bouldin index, and Calinski-Harabasz scores respectively. All scores are averaged over 10 runs.

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY