



CHALMERS
UNIVERSITY OF TECHNOLOGY



UNIVERSITY OF GOTHENBURG

From Data to Descriptions: Efficient Data Retrieval in Autonomous Vehicle Development using Generative AI

A Design Science Study

Master's thesis in Computer science and engineering

**JESPER KNAPP
KLAS MOBERG**

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2024

MASTER'S THESIS 2024

**From Data to Descriptions:
Efficient Data Retrieval in Autonomous
Vehicle Development using Generative AI**

A Design Science Study

JESPER KNAPP
KLAS MOBERG



UNIVERSITY OF
GOTHENBURG



CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2024

From Data to Descriptions: Efficient Data Retrieval in Autonomous Vehicle Development Using Generative AI
A Design Science Study
JESPER KNAPP
KLAS MOBERG

© JESPER KNAPP, KLAS MOBERG 2024.

Supervisor: Mirosław Staron, Department of Computer Science and Engineering
Advisors: Yudi Chen, Yuchuan Jin, Rasmus Lundin, Zenseact
Examiner: Mats Granath, Department of Physics

Master's Thesis 2024
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in L^AT_EX
Gothenburg, Sweden 2024

From Data to Descriptions: Efficient Data Retrieval in Autonomous Vehicle Development Using Generative AI

A Design Science Study

JESPER KNAPP

KLAS MOBERG

Department of Computer Science and Engineering

Chalmers University of Technology and University of Gothenburg

Abstract

The software that enables autonomous driving requires a variety of sensors that generate a large amount of data. Data collected from a vehicle are often stored for later reference, either for testing new software components or to analyze the fleet of vehicles on a larger scale. Due to the large amount of varied data, finding a specific vehicle scenario in a collection of vehicle logs proves difficult. Current solutions mainly use SQL to query the database of logs, this solution does however require knowledge of both SQL and the specifics of the data that you are looking for. This thesis was carried out in collaboration with Zenseact, and aims to create an artifact called "Genius" to enable searching their logged vehicle data using natural language by applying generative AI to generate scenario descriptions. A scenario is a 30 second snippet of data from the vehicle logs that contain signals, which are the result of processed sensor data from the vehicles. Videos recorded from the roof of the vehicles complement the signal data. The descriptions are created in two parts, first images from the video are fed to LLaVA 1.5 7b, a multi-modal LLM that describes the scenario based on the image. A selection of key signals extracted from each log, as all signals cannot fit inside the context size of the deployed LLM, are then combined along with the image description and fed to a second LLM, Gemma 7b, to create a combined description. After the descriptions have been generated they are embedded using BGE-large, a text embedding model, and stored in ChromaDB to create a vector database. This database is then used to allow semantically searching the logs by comparing their distances in vector space to a natural language query.

This study follows the design science research (DSR) methodology with three regulative cycles, with 5 phases in each, followed by a *learnings* section for each cycle with insights that are used in the subsequent cycle. Initial results with a smaller set of 8 scenarios show promising results in terms of how well the scenarios were separated in vector space, and the ability to search them using natural language. When scaling up to 100 scenarios, scenarios are mostly still searchable, however, scenarios that are not very distinct are hard to find since there are many similar matches. To counteract this, several systems of evaluating if the returned scenarios are correct were implemented, such as comparing keywords and an evaluation of the scenario distances. The generated descriptions were evaluated by engineers working with the vehicle logs at the collaborating company, on a scale of 1-5, the descriptions achieved a mean score of 3.3125. Overall, the solution can not replace existing solutions in its current form, this is due to the fact that all data is not available in the generated descriptions and the LLM and embedding models limited capabilities

with numerical data.

Keywords: Large Language Model, Generative Pre-trained Transformer, Semantic search, Embedding, Big data, vehicle logs, autonomous driving.

Acknowledgements

We would like to thank Zenseact for giving us the opportunity to pursue our thesis at their office and providing us with all the required resources. A special thanks to our supervisors at Zenseact: Yuchuan Jin, Yudi Chen and Rasmus Lundin, and our academic supervisor Mirosław Staron for their constant support and advice throughout our work. We would also like to thank Victor Bergström, who worked on a thesis adjacent to ours, for his collaborative spirit and for the constructive discussions that helped shape our thesis.

Jesper Knapp and Klas Moberg, Gothenburg, 2024-06-06



Contents

List of Figures	xv
List of Tables	xvii
1 Introduction	1
1.1 Problem statement and Purpose of the Study	2
1.2 Research Questions	3
1.3 Limitations	4
1.4 Significance of the Study	4
1.5 Structure of the Thesis	5
1.6 Related Work	5
2 Background	7
2.1 Natural Language Processing (NLP)	7
2.1.1 Tokenization	7
2.1.2 Embeddings	8
2.2 Large Language Models (LLMs)	10
2.2.1 Multimodal models	12
2.2.2 Prompting	13
2.2.3 Quantization	14
2.3 The dataset	15
2.4 t-distributed Stochastic Neighbor Embedding (t-SNE)	16
2.5 Word2Vec	17
3 Research design	19
3.1 Design Science Research	19
4 The Artifact	23
4.1 Models	23
4.1.1 Multi-modal model (LLaVA-1.5-7b)	23
4.1.2 LLM (Gemma-7b)	23
4.1.3 Embedding model (BGE-large)	24
4.2 Deployment of the models	24
4.3 Vector database (ChromaDB)	25
4.4 Scenario processing	25
4.5 Interface	26

5	Findings	29
5.1	Cycle 0 Findings	29
5.1.1	Problem	29
5.1.2	Solution Candidates	29
5.1.3	Validation	30
5.1.4	Implementation	30
5.1.5	Evaluation	31
5.1.6	Cycle 0 Learnings	32
5.2	Cycle 1 Findings	33
5.2.1	Problem	33
5.2.2	Solution Candidates	33
5.2.3	Validation	36
5.2.4	Implementation	37
5.2.4.1	Artifact Implementation	37
5.2.4.2	Implementation of the LLM model evaluation	39
5.2.5	Evaluation	44
5.2.5.1	LLM model evaluation	44
5.2.5.2	Artifact Evaluation	45
5.2.6	Cycle 1 Learnings	50
5.3	Cycle 2 Findings	50
5.3.1	Problem	50
5.3.2	Solution Candidates	51
5.3.3	Validation	53
5.3.4	Implementation	54
5.3.5	Evaluation	55
6	Conclusion	63
6.1	Discussion of the results	63
6.1.1	RQ1	63
6.1.2	RQ2	64
6.1.3	RQ3	64
6.1.4	Artifact discussion	65
6.2	Summary	66
6.3	Future Work	67
	Bibliography	69
A	Appendix	I
A.1	Full descriptions of scenario for model evaluation	I
A.1.1	Mistral 7b	I
A.1.2	Starling 7b	I
A.1.3	Gemma 7b	I
A.1.4	Gemma 2b	II
A.2	Prompts	II
A.2.1	Standard prompts	II
A.2.1.1	Multi-modal LLM Prompt (LLaVA 1.5 7b)	II
A.2.1.2	LLM Prompt (Gemma 7b)	II

A.2.2	Few-shot prompt (Gemma 7b)	II
A.3	Evaluation form	V
A.4	Queries and results	VIII
A.4.1	Queries with correct answers:	VIII
A.4.2	Queries with no answers:	XI
A.4.3	Test Queries:	XIV

List of Figures

1.1	Processing of a scenario.	3
1.2	Example of the differences in existing SQL searching solution (left) and the usage the artifact in this thesis aims to offer (right).	3
2.1	Whitespace tokenization.	7
2.2	Two iterations of merging with the BPE tokenization algorithm.	8
2.3	t-SNE representation of embeddings for three categories of words describing vehicle scenarios.	9
2.4	Model architecture of the Transformer [17].	11
2.5	Figure of an example prompt.	13
2.6	Figure of an example few-shot prompt.	14
2.7	Illustration of vehicle log format with randomized data-points.	15
2.8	Examples of images captured in the vehicle from the Zenseact Open Dataset [25].	16
3.1	One regulative cycle of the Design Science Research methodology.	19
4.1	Processing pipeline for a scenario in Genius.	26
4.2	User interface for scenario querying.	27
5.1	Flowchart for the in-house LLM solution	31
5.2	Querying for a scenario with Genius.	35
5.3	The interface that allows the vector database to be queried, currently queried for a snowy scenario.	39
5.4	Histogram of the ratings of the generated scenarios.	40
5.5	Histogram of the ratings of the generated scenarios.	41
5.6	Histogram of the ratings of the generated scenarios.	41
5.7	Histogram of the ratings of the generated scenarios.	42
5.8	t-SNE visualization of the test scenario set descriptions when queried for a scenario containing snow.	45
5.9	Histogram of the distances between each of the 8 scenario descriptions generated by Gemma 7b	46
5.10	Histogram of the ratings of the generated scenarios.	46
5.11	Example of a Z-score threshold fitted to a curve of scenario distances.	52
5.12	Example of a regression line fitted to a curve of scenario distances.	53
5.13	Image of the interface querying for a change of lane, showcasing the keyword search with boldening and color labeling.	55

5.14	Histogram of the distances between each of the 100 generated scenario descriptions.	56
5.15	Plot representing the distances between a query and each of the embedded scenarios.	57
5.16	Table showing the measured results for the different distance metrics.	57
5.17	Plot representing the distances between a query and each of the embedded scenarios.	57
5.18	Table showing the measured results for the different distance metrics.	57
5.19	Plot representing the distances between a query and each of the embedded scenarios.	58
5.20	Table showing the measured results for the different distance metrics.	58
5.21	Plot representing the distances between a query and each of the embedded scenarios.	58
5.22	Table showing the measured results for the different distance metrics.	58
5.23	Plot illustrating all scenario distances for the set of correct scenarios.	59
5.24	Plot representing the distances between a query and the average distances of all scenarios in the set.	59
5.25	Plot illustrating all scenario distances for the set of incorrect scenarios.	60
5.26	Plot representing the distances between a query and the average distances of all scenarios in the set.	60
5.27	Plot illustrating all scenario distances for the query test set.	61
5.28	Plot representing the distances between a query and the average distances of all scenarios in the query test set.	61
A.1	Evaluation form page 1	VI
A.2	Evaluation form page 2	VII

List of Tables

2.1	Document Frequencies (number of documents a term appears in) and Inverse Document Frequencies from the corpus of 8 descriptions of vehicle scenarios.	9
2.2	TF-IDF values for four words from two different documents in the corpus.	10
2.3	Sizes of Llama2 models in FP16 and INT4 formats.	14
3.1	Guidelines for Design Science Research by Hevner [28] and their application in this context.	20
4.1	Description of functions accessible through the API.	25
5.1	Measured call speed using a quantized Mistral 7b model with 20 samples.	32
5.2	Table of comparison metrics for each model. The best performing metric for each category is underlined. Metrics when outliers are excluded are enclosed in parentheses.	42
5.3	Table of different qualitative comparison metrics for each model.	44
5.4	Description of the rating scale in the form.	47
5.5	Comments on Question 2/6: "Are there any factual errors in the description of the scenario?".	48
5.6	Comments on Question 3/7: "Is there anything important missing from the description? For instance, any specific signals missing that you think would improve the description.".	49
5.7	Comments on Question 4/8: "Other comments.".	49
5.8	Table showing the average measured results for the different distance metrics measured from the set of correct scenario answers.	59
5.9	Table showing the average measured results for the different distance metrics measured from the set of correct scenario answers.	60
5.10	Table showing the average measured results for the different distance metrics measured from the query test set.	61

1

Introduction

The amount of software in the modern car is continually growing. One of the major drivers of this development in the automotive industry is autonomous functions. As software advances in autonomous capability, moving from features such as adaptive cruise control and assisted parking toward complete autonomy, the need for increasingly sophisticated software increases [1].

The many systems required to make autonomous vehicles a reality, generate a large amount of data that need to be processed and stored at a high rate [2]. As autonomous vehicles evolve, this data generation will continue to accelerate. Modern high-end vehicles are equipped with multiple imaging sensors for tasks like object recognition. However, autonomous vehicles usually require several different types of sensors: cameras, LIDAR, and radar. In addition, autonomous vehicles will need higher resolution images to ensure precision [3]. According to an article by IBM, cameras alone in Nvidia testing vehicles generate 1 Terabyte of data per hour [4].

An issue arises when trying to find and retrieve data from a particular scenario in the extensive database generated from sensors in the vehicles. Existing solutions are often based on querying a database using SQL (Structured Query Language), a programming language used for managing and manipulating databases. It allows users to retrieve, update, and delete data from a database [5]. However, this approach has some drawbacks in a scenario like this due to the volume and nature of the data, which includes varied signals from the vehicle and images with information of the scenario that are not searchable using SQL. To be able to find a specific scenario, the user needs domain-specific knowledge of the vehicle data and knowledge of SQL. Although this might be manageable for a technically involved engineer working with the data, it still takes time to figure out how to write the query. Furthermore, for a non-technical person, the insight that can be gained from the data might not be accessible at all due to the knowledge barrier.

This thesis was carried out in partnership with Zenseact, whose main goal is to develop autonomous software for Volvo Cars. Their fleet of test vehicles collects and stores data that is later processed to create logs with thousands of signals for each drive. The objective of the thesis is to use generative AI to query the vehicle log database for a specific scenario using natural language, in order to simplify the process of searching and understanding vehicle logs and merging information from both signals and images.

1.1 Problem statement and Purpose of the Study

The collaborating company's fleet of test vehicles is intended for data collection and come equipped with numerous sensors and cameras. All of these sensors and cameras generate high-frequency data that are then processed and combined to form a large set of signals that deliver information to several key systems in the car, such as automatic emergency brakes or adaptive cruise control. This set of signals are saved in a large table for each drive with a vehicle.

When developers at the collaborating company want to test edge cases of the software they are developing, for instance, if they are looking for a scenario where the vehicle is entering a tunnel or when the car passes an off ramp on the highway, the saved logs can be used to simulate how the car would behave in the real world. Problems arise when a software developer wants to find a specific scenario to test their code. Currently, finding the desired scenario requires them to write a complicated SQL query and then to query the combined database of all the saved logs. This not only requires knowledge of SQL and the structure of the database, it also requires knowledge of what signals are relevant for this specific scenario and how they behave.

This thesis aims to address the challenges that arise when searching for scenarios by exploring and implementing a solution using Generative AI to enable searching for scenarios with natural language, thus avoiding the process of writing complex SQL queries.

The following challenges are addressed in this thesis:

- (i) There are a large number of signals and it can be hard to know which signals are relevant.
- (ii) The signals are updated at high frequency, resulting in large amounts of data for each signal.
- (iii) You currently need knowledge of SQL to be able to query the logs, which might make them inaccessible for non-technical staff.
- (iv) Writing effective SQL code to find scenarios takes time.

Figure 1.1 visualizes the plan for processing scenarios. We define a scenario to be 30 seconds long and consist of both the signal data from the vehicle during that time and video from a camera mounted on top of the vehicles roof. The output from the pre-processing should be a text description that merges information from both the table of signals and the video.

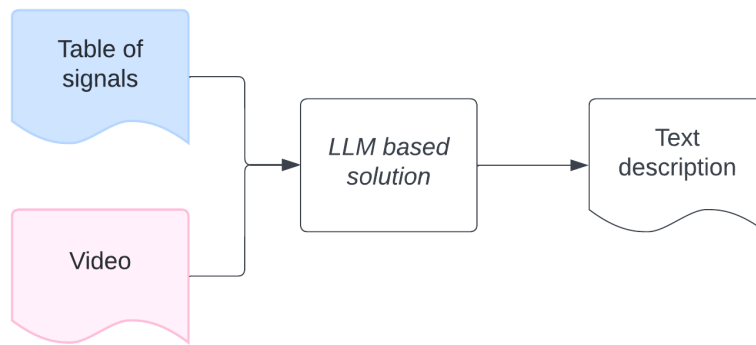


Figure 1.1: Processing of a scenario.

Using the generated text descriptions of the scenarios, the aim is to facilitate searching with natural language instead of SQL by using a vector database. Figure 1.2 shows how the solution would compare with the current workflow.

SQL Query

```

1 country_code = "752"
2
3 df = duckdb.query(
4     f"""
5     SELECT
6         d.vehicle,
7         d.utc_start,
8         d.utc_end
9     FROM
10    read_parquet('parquet_files/*.parquet.gzip') as d
11    WHERE
12    d."data/tunnel/distance_to_tunnel_entry" < 50
13    AND d."data/tunnel/has_upcoming_tunnel_entry" = 1
14    AND d."data/country/iso_code" = {country_code}
15    """
16 ).fetchdf()
  
```

Natural Language Query

"A scenario where the vehicle is entering a tunnel in Sweden."

Figure 1.2: Example of the differences in existing SQL searching solution (left) and the usage the artifact in this thesis aims to offer (right).

1.2 Research Questions

- **RQ1:** *To what extent can an LLM generate a description based on signals in the form of tabular data?* This is an investigation of how well an LLM can be used to summarize the specific scenario in which a vehicle is in and what steps need to be taken to improve the accuracy of the descriptions.
- **RQ2:** *To what extent can a multi-modal LLM with vision describe the scenario a vehicle is in based on images?* We want to explore how well an LLM with the ability to process images can qualitatively explain what scenario a vehicle is in based on images captured from the scenario.
- **RQ3:** *To what degree can a combination of descriptions from tabular and image data be used to capture the semantics of a vehicle scenario in a vector*

representation? We want to investigate how well the semantics of the descriptions can be represented as a vector in terms of distinguishing scenarios from each other; this will, in turn, be used to allow semantically searching for a scenario.

1.3 Limitations

The context of the explored solution will be limited to the data provided by the collaborating company. No investigation in the solutions effectiveness in other domains of data will be performed.

Specifically for the technical solution, due to the limited context size of language models, not all signals will be able to be utilized for generating descriptions. Instead, a subset of "key signals" will be selected, focusing on information relevant to the most common scenarios, these are chosen in collaboration with the supervisors at the collaborating company. Scenario descriptions are also limited in time resolution, scenarios will be chosen to have a specific length and a description will be generated that captures that length of time, therefore dynamics much shorter than the selected scenario length might be lost. The thesis is restricted to employing open-source models running locally, driven by considerations of cost and data security. Due to hardware restrictions, specifically in terms of video memory, the thesis is limited to using quantized models of smaller parameter sizes. All models with 7 billion parameters are quantized to 4-bit resolution, while smaller models can be quantized to 8-bit resolution.

One approach to generating text descriptions for a specific domain is fine-tuning a large language model. That approach requires training with labeled data and very large computational resources, however, no labeled data is available nor the required hardware. If an approach with training were to be investigated, the data would have to be manually labeled, which is a time-consuming and costly process that hinders the pace of development. Considering these factors, a fine-tuning approach won't be explored in this thesis.

Furthermore, research is limited to not developing a fully automatic system for converting existing tabular and video data to text generated descriptions. There is no control over the execution and configuration of the tools explored in this thesis at the company.

1.4 Significance of the Study

The goal of this thesis is to provide valuable information on how to generate and optimize semantic descriptions of tabular and image data, with the goal of semantically searching the data using natural language.

A successful implementation would lead to several benefits for employees of the collaborating company. A tool that can search for scenarios using natural language instead of SQL could save precious time for developers who want to test their code

for specific edge cases. It could also enable employees without domain-specific knowledge of vehicle signals or even non-technical employees to be able to work with the vehicle log data. For example, management staff who want to gain insight into the vehicles or plan data collection. It could also provide cost savings in the form of time savings for engineers. Saving time for engineers could, in turn, lead to faster development of software for autonomous driving. Faster software development could ultimately lead to safer autonomous vehicles and reduce the risk of accidents on the road. The benefits are not limited to just the collaborating company, similar structures of data can be found in all domains of industry.

The work of this thesis could also be used to generate training data that can be used to train a more general model to understand scenarios in tabular data. Implementing reliable auto-generated training data could lead to a great range of future applications. Fast auto-generation of labels could allow for quicker and inexpensive development, not only for models proficient in understanding tabular data, but also facilitate the swift and cost-effective development of a diverse array of machine learning models beyond those focused solely on table data.

1.5 Structure of the Thesis

This thesis will follow this structure: Chapter 1 introduces the subject and the challenges the thesis aims to address. In addition to that, Chapter 1 will also include the research questions, purpose, limitations, and related work. Chapter 2 covers the background knowledge of the technologies used during the thesis. Chapter 3 will focus on the research methodology that this thesis has followed. Chapter 4 will include a technical description of the artifact developed during the thesis. Chapter 5 presents the methodology of the thesis in the form of findings that are split over three cycles. Finally Chapter 6 will discuss the findings of the thesis, ending with a conclusion and how the work could be continued in the future.

1.6 Related Work

Many different implementations of table-to-text generation have been explored previously. One example of table-to-text generation is the generation of biographies using Wikipedia information tables, where the authors trained a language model to output a biography from an input table of information for the person [6]. The technique used to extract information from each field of the table and generate a coherent description is similar to our task, since we are extracting data from columns that correspond to specific signals.

Another example, titled *Investigating Table-to-Text Generation Capabilities of LLMs in Real-World Information Seeking Scenarios* [7], explores how well different existing models can answer questions based on information in tabular data, including LLaMA2, GPT-3.5, and GPT-4. The experiments were done using several different table datasets and, based on their results, only GPT-3.5 and GPT-4 could generate reasonable results without a few-shot prompt, where the prompt to the model in-

cludes examples of the task. Furthermore, the results show that for some datasets, small models have a comparative performance to large models in understanding of the tables. Since we are limited to using smaller models in this thesis, this result is promising.

A more recent Microsoft article presents *TableGPT*, an LLM that has been trained to complete a variety of tasks in structured data, one of which is to summarize the contents of a table. In their approach, they use instruction fine-tuning to improve the performance of GPT-3.5 on tabular input, resulting in a substantial improvement in table tasks such as column extraction and error detection[8]. The problem with their approach to tuning the LLM is that the entire table has to fit in the instruction prompt, something that is not feasible for our application since the prompt then would have to fit tokens in the order of billions. This fact is also true for the earlier reviewed paper, where different models were compared; the approach hinges on that the whole table can be fit inside the prompt. However, if the goal is a tool that has a real-world impact, a model in which the table must fit the prompt is rarely enough. The work of analyzing tables often focus on tables that are very large, so the usefulness of such a model would be greatly improved if larger tables could be processed.

The limitation of the previous studies discussed in this section is that the proposed solutions, as mentioned above, are suitable only for tables of a smaller scale compared to the data in this thesis. Additionally, a distinction lies in the fact that the models in those studies are trained on tables that include both text and numerical information, whereas this thesis will concentrate solely on numerical data.

The concept of using vehicle-gathered data to explain and label scenarios is not entirely groundbreaking. Existing work, exemplified by *Automatic Labeling to Generate Training Data for Online LiDAR-based Moving Object Segmentation* [9], focuses on harnessing data from test drives-specifically LiDAR signals-to produce labels for training data, for which they reported successful results. In this context, the labels take the form of images with marked points of interest. Although this approach diverges from the methodology and outcomes of the present thesis, which revolves around generating textual descriptions using LLMs and vision models, there is a shared objective. Both endeavors aim to generate labels that facilitate future machine learning applications, underscoring the industry’s demand for the automated generation of reliable labels. The difference being that this thesis aims to generate more broadly applicable labels in the form of textual descriptions instead of specifically labeled images.

2

Background

In this chapter, essential background information relevant for the thesis is presented. The discussion will include the following topics: Natural Language Processing, Large Language Models, Multimodal Models, Quantization, and Embeddings. This chapter serves as the theoretical foundation for the thesis.

2.1 Natural Language Processing (NLP)

The field of NLP has been studied since long before modern Large Language Models were ever imagined. Alan Turing is considered to have sparked the research in the field in the 1950s when he introduced the *Turing Test*, which aimed to decide if a machine was discernible from a human. The focus of NLP is to bridge the gap between human language and the language of computers [10].

2.1.1 Tokenization

One of the most essential concept of NLP is tokenization; it is the first step in converting natural language to be able to be processed by a mathematical model. In the English language, words are often split by white spaces, but that might not always be the most accurate, the meaning of some words rely on them being read together, such as *Los Angeles* or *Machine learning*. There are many different methods of tokenization, varying in complexity and benefit depending on the task. The simplest form of tokenization is to simply split all words at a whitespace character, which is enough detail for some applications.

"The cat is sleeping" \longrightarrow ["The", "cat", "is", "sleeping"]

Figure 2.1: Whitespace tokenization.

More advanced tokenization models can tokenize a word into several tokens, so-called sub-word tokenization. An example of such a tokenizer is the Byte-Pair Encoding (BPE) tokenizer [11]. The way BPE works is by first finding the set of all unique words in the text corpus (corpus refers to a structured collection of text) and building a vocabulary of all the symbols that are necessary to write those words; these are the tokens. After that initial vocabulary of tokens is built, new tokens are added by *merges*, which means combining two tokens into one token and thus creating

longer tokens. The tokens that are most commonly found together in the corpus are merged in each step, and this continues until the vocabulary reaches a desired size. Tokenizing on sub-words could lead to the model being able to understand words not seen before or to connect parts of words together and understand them [12]. This particular algorithm was the choice for several advanced LLMs a few years back, such as GPT and GPT-2 [13].

If we assume that we have the same words as in the previous example, with the following frequencies in our corpus. We begin by splitting each word into characters and denoting their frequencies. Our starting vocabulary is the unique characters from the list of words. Figure 2.2 shows two iterations of merging using BPE.

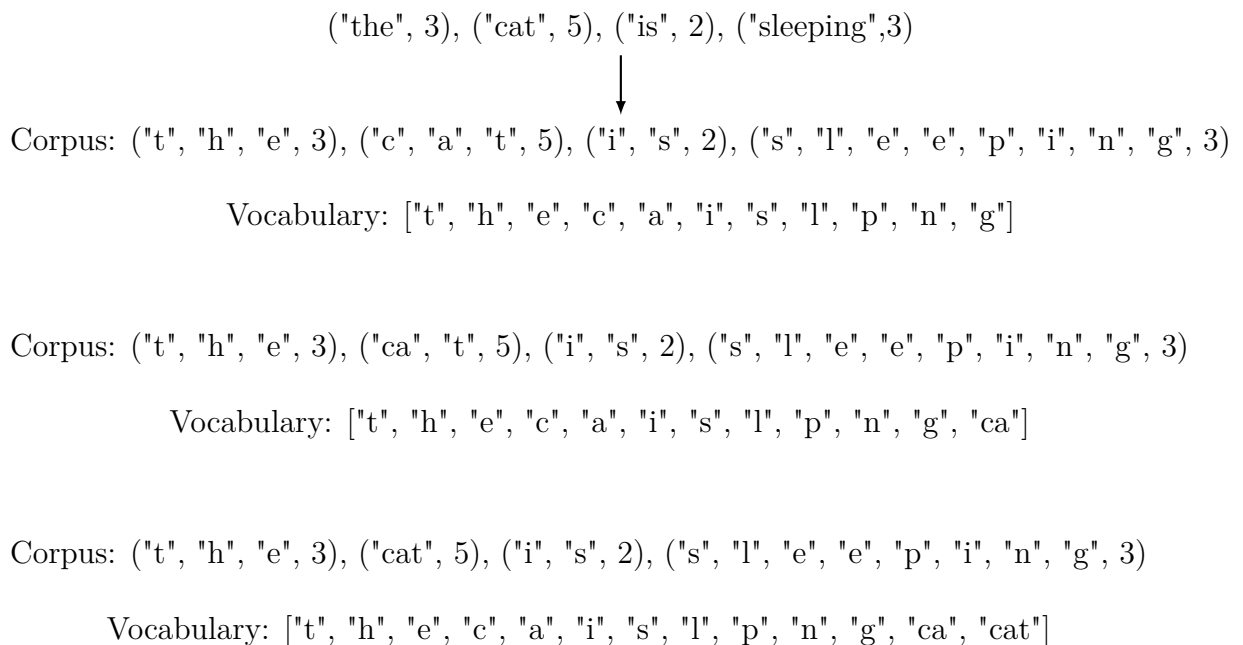


Figure 2.2: Two iterations of merging with the BPE tokenization algorithm.

2.1.2 Embeddings

Another key concept in the field of NLP is embeddings, these serve as a vector representation of text in a continuous vector space. Just like tokenization, there exists many different methods and models to embed text ranging in complexity. These representations enable machines to grasp the subtleties of human language, mimicking the way humans comprehend linguistic nuances [14]. An illustration of how embeddings can capture the meaning of different categories of words are shown in Figure 2.3.

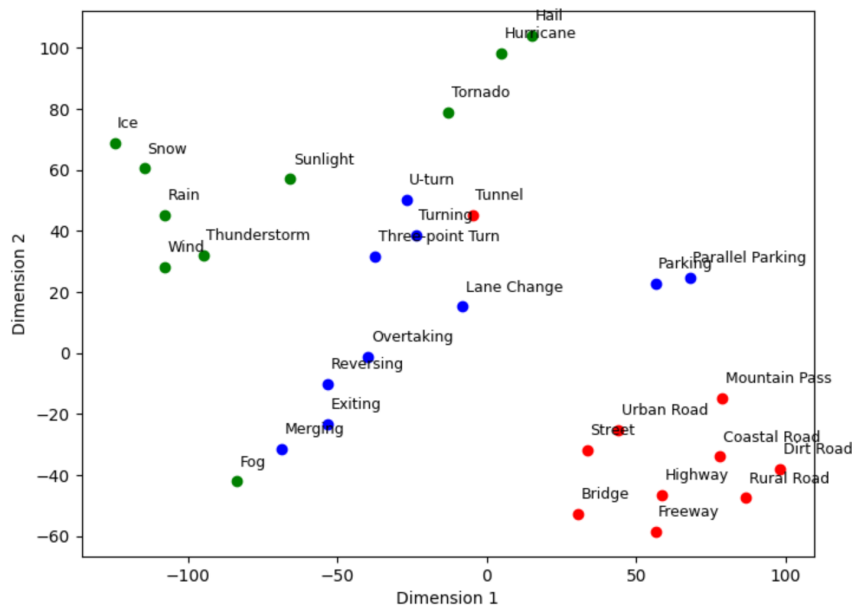


Figure 2.3: t-SNE representation of embeddings for three categories of words describing vehicle scenarios.

An example of a simple embedding model is TF-IDF (Term Frequency-Inverse Document Frequency) that use the frequency of terms in a corpus to try and capture what words are important and set weights to each of the terms based on their respective importance. In this context, a Document refers to one piece of text and a Corpus refers to a collection of Documents. TF-IDF weighting is widely used for recommender systems, according to a survey from 2015, 83% of text-based recommender systems were based on TF-IDF [15]. The TF-IDF weights are calculated as follows.

$$\text{TF}(t, d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d}$$

$$\text{IDF}(t, D) = \log \left(\frac{\text{Total number of documents in the corpus } D}{\text{Number of documents containing term } t} \right)$$

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \cdot \text{IDF}(t, D)$$

Term	DF	IDF
Austria	1	2.39
clear	1	2.39
America	2	1.98
weather	3	1.69
events	4	1.47
appears	5	1.29
visible	6	1.13
no	7	1.00
trend	8	0.88
velocity	8	0.88

Table 2.1: Document Frequencies (number of documents a term appears in) and Inverse Document Frequencies from the corpus of 8 descriptions of vehicle scenarios.

Term	Document 1	Document 3
with	0.1319	0.0000
driving	0.0971	0.0929
highway	0.0000	0.0738
maximum	0.0510	0.0000

Table 2.2: TF-IDF values for four words from two different documents in the corpus.

In Figure 2.1 we can see that words *trend* and *velocity* appear in all the scenarios so those words does not give any relevant information, so therefore their weights will be low. In contrast, the word *Austria* only appear in one scenario and that is therefore a more important word.

More advanced embedding models aim to capture more features of natural language, such embeddings enable the model to recognize relationships in the semantic vector space, such as subtracting "Man" from "King" and adding "Woman" to derive "Queen".

Similarity is another important concept, it seeks to show how semantically related two words or texts are based on the distance between their respective vectors. Two common ways to calculate this distance are the Squared Euclidean (Equation 2.1) distance and the Cosine Similarity (Equation 2.2) between vectors \vec{A} and \vec{B} .

$$d = \sum_i (\vec{A}_i - \vec{B}_i)^2 \quad (2.1)$$

$$d = 1.0 - \frac{\vec{A} \cdot \vec{B}}{|\vec{A}| |\vec{B}|} \quad (2.2)$$

2.2 Large Language Models (LLMs)

LLMs represent a significant leap in natural language processing and artificial intelligence. Unlike traditional rule-based systems, these machine learning models, driven by deep learning techniques, excel at comprehending and generating extensive human-like text.

As the name suggests, LLMs are large, often comprising of billions or even trillions of parameters. For example, GPT-4, one of the current state-of-the-art models, is believed to consist of 1.76 trillion parameters [16]. Parameters refer to the number of trained weights inside model. The sheer size enables these models to capture subtle nuances and intricate linguistic patterns present in large language datasets.

The most important component in the architecture of a modern LLM is the transformer. Introduced in "*Attention is all you need*" in 2017 [17], it set the foundation for the advances seen during the recent years in the field of LLMs. The transformer model allows models to generate coherent responses based on a large amount of text information at a scale not possible before [12].

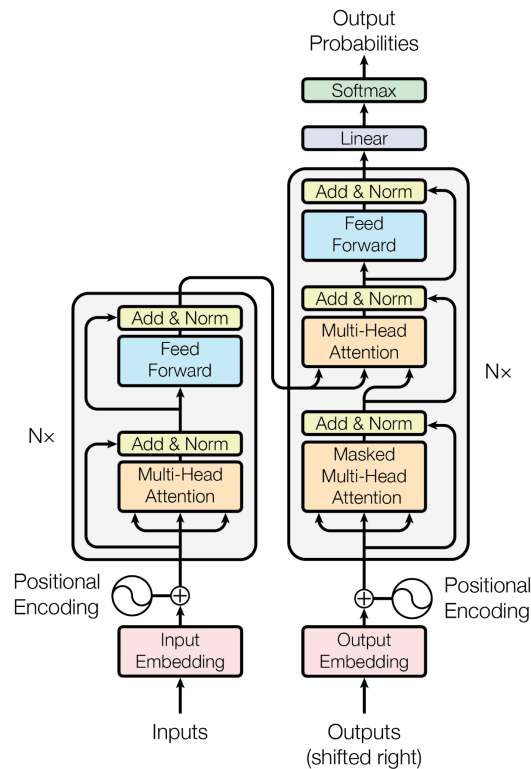


Figure 2.4: Model architecture of the Transformer [17].

The revolutionary part of the transformer architecture is the *attention mechanism*, which can be thought of as the model being able to "pay attention" to specific parts of a long text during processing and generation. The attention mechanism is what enables the models to extract information from arbitrarily large contexts, and allows it to capture long range dependencies in the data in a way not seen before [10].

The transformer architecture is based on two types of layers, encoders and decoders. The encoders main goal is to capture all relevant information from the input and create a representation of that information that later can be passed to the decoder. The way this is achieved is through using NLP techniques such as tokenization to process the input text. The encoder also adds a positional component to the representation by using cosine functions of different frequencies.

The decoder generates output based on a given input. It takes the output of the encoder and generates output tokens, one at a time. Both the encoder and decoder makes use of the attention mechanism to understand the relationships between tokens.

A transformers based LLM is trained in two main steps, advanced models might consist of several more specific steps that increase the performance, but those might not always be disclosed. The first step is pre-training, where the model learns about language and information about the world using a large amount of text data. After this step the model acts as a auto complete model, if prompted with, for example, "The cat is", it might generate the text "sleeping." [12]. To get the model to follow

prompted instructions, the model undergoes so called instruction-tuning. In this stage the model is fed a large set of instructions along with the desired output to learn how to follow them.

The transformer is the underlying model for all state-of-the art LLMs since its introduction, one of the first major LLMs that utilized the transformer architecture was BERT, released by Google in 2018 it achieved new state-of-the art results on a wide range of language tasks, such as question answering and language inference [18]. Another big advancement was made by OpenAI in 2020 when they released their GPT-3 model that surpassed the previous benchmarks set using a record model size of 175 billion parameters [19], in comparison, the largest version of the BERT model only comprised of 185 million parameters.

LLMs started to gain a lot more widespread recognition following the release of OpenAI's ChatGPT in late 2022 [20], which is an LLM that can be accessed online and powered by an iteration of GPT-3 called GPT-3.5. OpenAI has also later released yet another iteration called GPT-4, a so called multimodal model, that can accept both text and images as input [16].

The practical applications of LLMs span various domains, making them essential for tasks such as natural language understanding and text generation. The different uses of LLMs are currently being widely researched and are changing many industry standards at the time of writing.

2.2.1 Multimodal models

Multimodal models [21] are LLMs that are capable of more than one mode of communication, in this context language and visual understanding. Similar to conventional LLMs, vision encoding models operate in a different mode by also accepting images as input, which are then encoded and decoded to generate a relevant text response corresponding to the visual content. For most contemporary multimodal models, a dual input approach is common, incorporating both a text input and an image. This integration allows for additional instructions on how to process the image, such as describing its contents. This dual-input mechanism enhances the versatility of vision models, enabling them to perform tasks that involve both visual understanding and language processing.

Multimodal models capable of both textual and visual understanding use the standard LLM architecture discussed above for the textual understanding. For visual understanding however, Convolutional Neural Networks (CNN) comes into play.

CNNs are a class of deep neural networks particularly suited for tasks involving visual data, such as image recognition or classification. Key components that constitute the CNN structure include:

1. **Convolutional Layers:** These layers consist of filters known as kernels that slide over the input image, performing convolution operations. These oper-

ations help extract features from the images by capturing spatial patterns. Each kernel is specialized to detect specific features such as edges, textures or shapes. This results in the generation of a feature map, a two-dimensional representation of the kernels generated outputs.

2. **Pooling Layers:** Pooling layers downsample the feature maps, reducing their spatial dimensions. Common pooling operations include max pooling and average pooling, which help to retain the most vital information while reducing computational complexity.
3. **Activation Functions:** Activation functions introduce a non-linearity into the network, allowing CNNs to learn complex relationships in the data. Common activation functions used in CNNs include Rectified Linear Unit (ReLU), sigmoid, and tanh.
4. **Fully Connected Layers:** Also known as dense layers, Fully connected layers are typically added towards the end of the architecture. These layers perform classification or regression tasks based on the features extracted by earlier layers, connecting every neuron from the previous layer to every neuron in the subsequent layer.

2.2.2 Prompting

In the context of LLMs, a "prompt" is the input to the model, often in the form of text, that serves as an instruction for the model. Prompting is the technique of tweaking the input prompt in order to get a desirable output from the model. Typically, prompts just contain the question the user wants answered, however it might sometimes contain supplementary information. This information could be in the form of a desired output format, background information etc. For example, a company-specific chatbot might receive prompts that contain unique details about the organization.

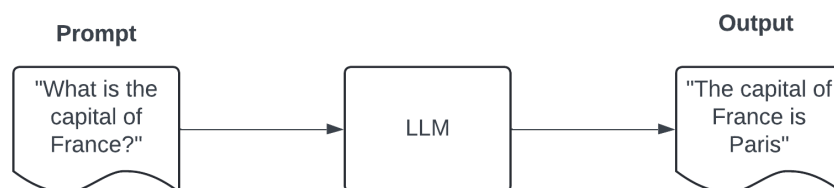


Figure 2.5: Figure of an example prompt.

A more advanced prompting technique is known as few-shot prompting [22]. Few-shot prompting involves presenting a sequence of examples of inputs and desired outputs to the model. This method aims to improve both accuracy and to ensure that the desired output format is adhered to.

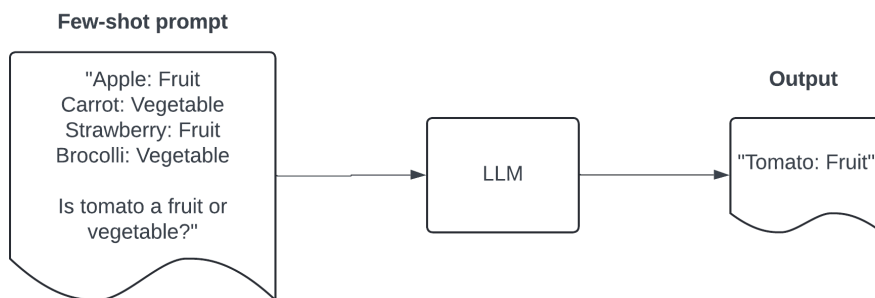


Figure 2.6: Figure of an example few-shot prompt.

2.2.3 Quantization

Model quantization is an approach employed to decrease the size of expansive neural networks, such as LLMs, by adjusting the precision of their weights. This method involves reducing the resolution of the model weights by decreasing the number of bits utilized.

There are two types of LLM quantization:

1. Post-Training Quantization (PTQ) [23]: Converting the weights of a trained model without retraining. Simple and easy to implement.
2. Quantization-Aware Training (QAT) [24]: Integrates weight quantization during the training process. This usually leads to a more accurate model but needs to be considered before the training process.

Quantizing a model has a significant effect on the required memory usage; for example, Table 2.3 shows the effect of quantizing the three Llama2 models from using floating point with 16 bits for the weights to using integers with 4 bits.

Model	Original Size (FP16)	Quantized Size (INT4)
Llama2-7B	13.5 GB	3.9 GB
Llama2-13B	26.1 GB	7.3 GB
Llama2-70B	138 GB	40.7 GB

Table 2.3: Sizes of Llama2 models in FP16 and INT4 formats.

Many quantized models are readily available for download on Hugging face. Most popular models have many quantized versions available.

Surprisingly, quantization does not always lead to lower accuracy, although most often it does. Meta reports that in some cases the accuracy actually increases, while also boasting lower latency and enhanced throughput [24].

2.3 The dataset

The data used in this thesis are collected from test drives conducted by the collaborating company. During these test drives, the vehicles collect and store data from a variety of sensors. Thereafter, the collected data undergo multiple software processes, which results in a table comprising several thousand (~ 6000) processed signals. Some examples of signals are whether there is snow in the lane in front of the vehicle or whether the vehicle is currently in a tunnel.

Each vehicle log corresponds to one drive with a test vehicle; the time span of each drive ranges from less than an hour up to full 8-hour days of driving. The processed table of signals contains the signal data from the entire duration of the drive. These tables are saved and accessed in parquet format. Since the table consists of thousands of signals and there are hundreds of thousands of data points for each signal, the table is on the order of 10^8 entries. Capturing what happens during the entire drive in one text would not be practical; therefore, the logs are split into "Scenarios". Each scenario is 30 seconds long and contains all the signals from the corresponding time. The resulting table for one scenario is instead on the order of 10^6 entries, which still makes traditional methods of feeding all the data to an LLM impossible.

Signals/Headers					
	Signal_1	Signal_2	Signal_3	Signal_4	Signal_5
0	0.905321	0.259464	0.842408	0.002313	0.387547
1	0.462026	0.385417	0.424411	0.133905	0.070351
2	0.263864	0.111750	0.046741	0.296309	0.521766
3	0.361231	0.839834	0.178092	0.947370	0.317423
4	0.925982	0.890872	0.601594	0.603258	0.944391
...
95	0.051028	0.122059	0.058176	0.288759	0.845617
96	0.227059	0.388239	0.560617	0.352162	0.924338
97	0.541383	0.617868	0.188423	0.926432	0.703087
98	0.869152	0.242536	0.532085	0.723787	0.325234
99	0.240387	0.114625	0.667535	0.052146	0.986356

frequency x 30 seconds

Figure 2.7: Illustration of vehicle log format with randomized data-points.

Complementing the table of signals is video captured from a camera mounted on top of the vehicle looking ahead. The videos are saved in mp4 format in 30 second intervals.



Figure 2.8: Examples of images captured in the vehicle from the Zenseact Open Dataset [25].

To be able to utilize both data sources, after the equivalent of 30 seconds of signal data has been extracted from the table, it has to be matched to the corresponding video. The snippet of signals matched with the video file is what makes up the data for each scenario. Some examples of vehicle scenarios explored in this thesis are: vehicle in a specific country, driving through a tunnel, and driving in snow.

2.4 t-distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE is a dimensionality reduction technique introduced by Laurens van der Maaten and Geoffrey Hinton in 2008 [26]. Unlike traditional methods such as Principal Component Analysis (PCA), which focus on preserving global structures, t-SNE emphasizes preserving local relationships in high-dimensional data. t-SNE employs the following key concepts as part of its structure.

Key Concepts:

1. **Similarity Measurement:** t-SNE computes pairwise similarities between data points using Gaussian distributions in the high-dimensional space, ensuring nearby points are considered similar.
2. **Student's t-Distribution:** t-SNE constructs a similar distribution in a lower-dimensional space using Student's t-distribution, alleviating crowding problems by providing heavier tails compared to a Gaussian distribution.
3. **Cost Function Optimization:** t-SNE minimizes the Kullback-Leibler (KL) divergence between the probability distributions, aiming to place similar points closer together in the lower-dimensional space.

Applications:

t-SNE is widely used in various fields, including natural language processing, image analysis, and bioinformatics. It aids in visualizing word embeddings, analyzing gene expression data, and identifying clusters with similar patterns.

Considerations:

While t-SNE provides intuitive visualizations, it has limitations, such as computational complexity and sensitivity to hyperparameters. It may not preserve global structures as effectively as PCA, leading to potential distortions in the visualization.

Despite these limitations, t-SNE remains valuable for exploratory data analysis, providing insight into the underlying structures of complex datasets.

2.5 Word2Vec

Word2Vec is a natural language processing technique to find vector representations of words created by Tomas Mikolov in 2013 [27]. Word2Vec consists of a group of shallow, two-layer neural network models, that work together to construct word embeddings. It operates on the principle that words appearing in similar contexts share semantic meaning. Word2Vec trains the neural networks to predict context words given a target word (Skip-gram) or predict a target word given its context (Continuous Bag of Words). By doing so, it learns dense vector representations of words that capture semantic relationships. In this thesis a Word2Vec model trained on the Google News dataset (~100 billion words) and the Skip-gram method will be used to find semantically similar words given a target keyword.

3

Research design

The research conducted in this thesis follows the Design Science Research (DSR) methodology using guidelines described in Hevner’s paper [28] and the definition of regulative cycles by Wirenga [29]. In the next section, we will present the methodology and how it was applied in this study.

3.1 Design Science Research

Design Science Research is an iterative method that aims to produce an artifact that solves or improves a problem in its context and to provide new information in its domain of knowledge [30]. One iteration, called a regulative cycle, consists of five steps, problem formulation, solution candidates, validation, implementation, and evaluation. The five steps are executed in all iterations, but the emphasis of the research should shift in each cycle and the information from the evaluation in the previous cycle should be used as input for the next cycle [29]. The five steps in each cycle can be seen in Figure 3.1.

An important aspect of DSR is the differentiation between conventional design and DSR. Hevner discusses this matter and asserts that there exists a disparity in the types of problem and solution between the two. Conventional design utilizes existing knowledge to address a problem and design a solution. In contrast, DSR aims to offer innovative solutions to unresolved issues and to enhance the existing knowledge base [31].



Figure 3.1: One regulative cycle of the Design Science Research methodology.

Guideline	Description	Application
Guideline 1: Design as an Artifact	Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.	The artifact in this thesis is the software tool that searches scenarios based on their generated descriptions, called "Genius".
Guideline 2: Problem Relevance	The objective of design-science research is to develop technology-based solutions to important and relevant business problems.	The problem our artifact addresses is the difficulties that a large database of vehicle logs pose when trying to search for a specific scenario. A solution to this problem has big relevance to both the collaborating company in particular, but also for other contexts with structured data searching.
Guideline 3: Design Evaluation	The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.	The performance of the solution is evaluated in terms of accuracy and speed both quantitatively and qualitatively.
Guideline 4: Research Contributions	Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.	The thesis aims to contribute to the knowledge base of LLM based systems design in similar problem contexts.
Guideline 5: Research Rigor	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.	The design of our artifact follows regulative cycles, each contains steps to define the problem, suggest solutions, validate the solutions with appropriate stakeholders, implement the solution and then rigorously evaluate the implementation to extract learning for our next cycle.
Guideline 6: Design as a Search Process	The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.	The artifact is developed using resources available in the collaborating company's infrastructure and is both designed and evaluated using data from their domain.
Guideline 7: Communication of Research	Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.	The research methodology background and application is presented in this chapter; in the Findings chapter, the methods and workflow of the research are described in detail.

Table 3.1: Guidelines for Design Science Research by Hevner [28] and their application in this context.

This thesis consists of three regulative cycles, and the produced artifact is a software tool named "Genius" that generates text descriptions of scenarios based on data from tabular vehicle logs and videos, and enables the user to semantically search the processed scenarios. The first cycle focuses on facilitating the infrastructure necessary to build our artifact, in particular deploying LLMs on a virtual machine at the collaborating company and implementing an API to be able to call the models. The first cycle ends with an evaluation of various open source LLMs that are candidates for the model in the final artifact; this evaluation consisted of both quantitative and qualitative parts. The second cycle shifts the focus to the main questions we are exploring in this thesis: How can LLMs create descriptions of scenarios based on tabular and image data? This cycle aims to better understand the problem and design a preliminary solution that contains all essential elements of the final artifact, followed by an evaluation. This cycle aims to answer **RQ1** and **RQ2**. The third and final cycle aims to increase the scale and improve the performance of the artifact using knowledge acquired from the second cycle, ending with a final evaluation. This evaluation aims to give the insight needed to answer **RQ3**. The following is a description of the five steps of each cycle.

1. **Problem**

The cycle starts with an investigation to provide a better understanding of the problem. Wirenga defines four separate reasons for investigating practical problems: problem-driven, goal-driven, solution-driven, and impact-driven [29]. The stakeholders of the collaborating company have the goal of providing a more intuitive solution for vehicle log searching; therefore, this is a goal-driven investigation. A goal-driven investigation of the problem in our case includes defining and prioritizing the goals to be achieved.

2. **Solution Candidates**

The second step is to generate possible solutions based on the findings of the first step. Generating a solution in this step does not mean to validate or implement it fully, but rather designing the idea of the solution to be able to communicate it to stakeholders. This is a creative step where many different paths are considered. The goal is to provide stakeholders with different possible solutions to help bridge the gap between the goal and where they are now. The outcome of this phase is an outlined plan that is to be validated and implemented.

3. **Validation**

Validation is a step in which stakeholders answers the question of whether or not the proposed design will bring them closer to their goal.

4. **Implementation**

The next step is to implement the validated design from the previous step. What this step implies varies greatly depending on the type of solution that is designed; in our case, it is the development of a software artifact. In the initial cycle (Cycle 0), the software framework is developed, followed by the creation of a small-scale solution in the subsequent cycle (Cycle 1), which includes the implementation of data processing steps and user communication

functionalities. Finally, Cycle 2 iterates on the same software artifact.

5. **Evaluation**

In this final step of the regulative cycle, the result of the previous step is evaluated to analyze whether the implementation addresses the problem investigated in the cycle. This is used as input for the problem step in the next regulative cycle.

4

The Artifact

The artifact is called "Genius" and consists of 5 components, the models, an API for accessing the models, a vector database, a processing pipeline for scenarios and an interface that interacts with the database and the models. This chapter describes them all in detail.

4.1 Models

This section describes the models we are using in our artifact in more detail, both the two different LLM's that are used for processing the scenario and also the embedding model that is used for semantically capturing the generated scenario descriptions.

4.1.1 Multi-modal model (LLaVA-1.5-7b)

Large Language-and-Vision Assistant (LLaVA) is a multimodal language model that achieves state-of-the-art performance on a wide range of benchmarks by training on GPT-4 generated language-image instructions. There are several model sizes available, due to the hardware limitations in terms of video memory, we chose to use the smallest model with 7 billion parameters. We chose the LLaVA model because of the performance and the availability of a version that is implemented using the huggingface transformers package, making for easier deployment on the VM since it functions the same as other LLMs tested throughout the thesis. It allows for prompts up to a size of 4096 tokens and handles images at a resolution of 336x336 pixels [21].

4.1.2 LLM (Gemma-7b)

Gemma is a lightweight family of open-source models developed by Google [32]. There are two model sizes available, with 2 billion and 7 billion parameters. After evaluating both sizes of Gemma, we concluded that Gemma 7b was the best model for our artifact. When running the 7b model, the weights are quantized to 4 bit resolution, when running the 2b model, we could instead run with 8 bit quantization, the higher resolution weights did, however, not outweigh the benefits of having a larger model in terms of parameters. The models are also available in two varieties of training, models that are only pre-trained, and models that are instruction-tuned. We chose to use the instruction-tuned models, since we are not fine-tuning the

model ourselves. The model has a token limit of 8192 tokens, which gives us plenty of room to employ few-shot prompting and to generate long descriptions; however, the quantization might limit the model's ability to comprehend long inputs and outputs.

4.1.3 Embedding model (BGE-large)

To be able to semantically compare generated descriptions of the scenarios with a user query, we use a text embedding model. To choose the most suitable model for our purposes, we used the Hugging Face Massive Text Embedding Benchmark (MTEB) Leaderboard [33], where embedding models are compared in several different benchmarks. The most important aspect for our artifact is the "retrieval" performance of the embedding model, how well it can compare a query to a collection of documents and retrieve the closest ones. In this metric, a model called BGE-large scores high on the benchmark, while also being relatively small in size with 335 million parameters. BGE is a general text embedding model part of C-pack, a package consisting of training data, benchmarks, and embedding models. It is available in several different sizes, and since they are not very computationally intensive, we chose the largest model, which also boasts the highest performance [34].

4.2 Deployment of the models

The models are stored on a local virtual machine at the collaborating company. The VM is equipped with a GPU with 16GB of VRAM to be able to run the models. The amount of VRAM limits us to using models with 7 billion parameters and also to lower the resolution of the weights in the models to 4 bit using quantization. The VM has both the LLMs stored (LLaVA 1.5 7b and Gemma 7b) and the embedding model (BGE-large), the LLMs are run on the GPU while the embedding model is run on the CPU. The embedding model would just like the LLMs have increased inference speed by running on the GPU instead of the CPU, but this setup allows us to run the two models simultaneously. The limited amount of VRAM leads to just one of the LLMs fitting in the memory at once, to be able to switch between them, you first need to unload one model and then load the other one.

To be able to access and infer with the models outside the VM, we built an API in collaboration with another thesis at the collaborating company. The API is based on the FastAPI package and the functions available are shown in Figure 4.1. The API does not facilitate transferring images to the multi-modal mode from a separate machine, instead the current implementation relies on the images that are to be used being uploaded to the VM beforehand, this was done to simplify the API and VM setup.

Name	Description
get_status	Returns basic information about the VM, if it is running, what model is loaded and the current time.
load_model	Loads a model to the GPU memory.
unload_model	Unloads a model from the GPU memory.
embed_text	Embeds the input string using BGE-large running on the CPU and returns the embedding vector.
query_llm	Queries a loaded LLM with the input prompt and returns the answer along with the total token count and the generation time. This requires an LLM to be loaded.
query_vision	Queries a loaded multi-modal LLM with the input prompt and returns the answer along with the total token count and the generation time. This requires an multi-modal LLM to be loaded.
find_similar_words	Takes a word as input and returns a list of the 25 semantically closest words using the word2vec model and the skip-gram method.

Table 4.1: Description of functions accessible through the API.

4.3 Vector database (ChromaDB)

After the scenarios have been processed and embedded, we store them in a vector database. For this, we have chosen to use ChromaDB since it is an open-source solution that is easy to implement. ChromaDB creates a local database, called a collection where the scenario descriptions can be stored with their respective embedding, and it also supports specifying metadata for each entry. We utilized metadata to store what vehicle, log file and the time of the scenario, this is useful when retrieving the scenarios since it makes it easier for users to find more data for the corresponding scenario. ChromaDB integrates with the local embedding model, when a scenario is added to the collection, an embedding is automatically created using the model and then stored in the collection. When you query for a scenario in the collection, ChromaDB also handles the vector similarity search, we chose to use squared Euclidean distance as the distance metric between embedded scenarios.

4.4 Scenario processing

The processing of each scenario follows the framework shown in Figure 4.1.

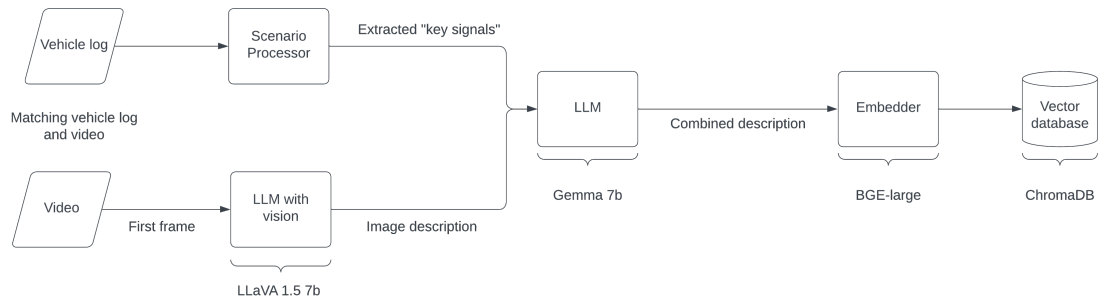


Figure 4.1: Processing pipeline for a scenario in Genius.

The information related to each scenario is organized in instances of a class called Scenario, where information such as vehicle, timestamp and the specific log file the scenario is based on is stored. The following process is followed for each of the scenarios: first each of the extracted video frames from each of the scenarios are processed using LLaVA-1.5-7b to generate a description of the image. Then each log file is processed to extract the selected signals, each signal is processed with a list of tools, such as the max, min or mean value, if it was activated or not and how long it was activated. This is decided for each of the signals individually beforehand, we are currently processing 16 signals from each log file. This information is then combined in a text string and fed to Gemma 7b model along with the image description that was generated earlier. To make the output format of the LLM more coherent, we employ a few-shot prompt. We feed the LLM 2 examples of how a final scenario should be structured. This output is then saved in the respective instance of the Scenario class. We also save a link to an internal visualization tool at the collaborating company in this class to make it easier to access more data from the vehicle scenario. Finally the complete scenarios are fed to our text embedding model (BGE-large) and added to the vector database in ChromaDB to make them searchable.

4.5 Interface

The interface for 'Genius' streamlines the querying and debugging processes for users. Illustrated in Figure 4.2, the interface is divided into distinct sections.

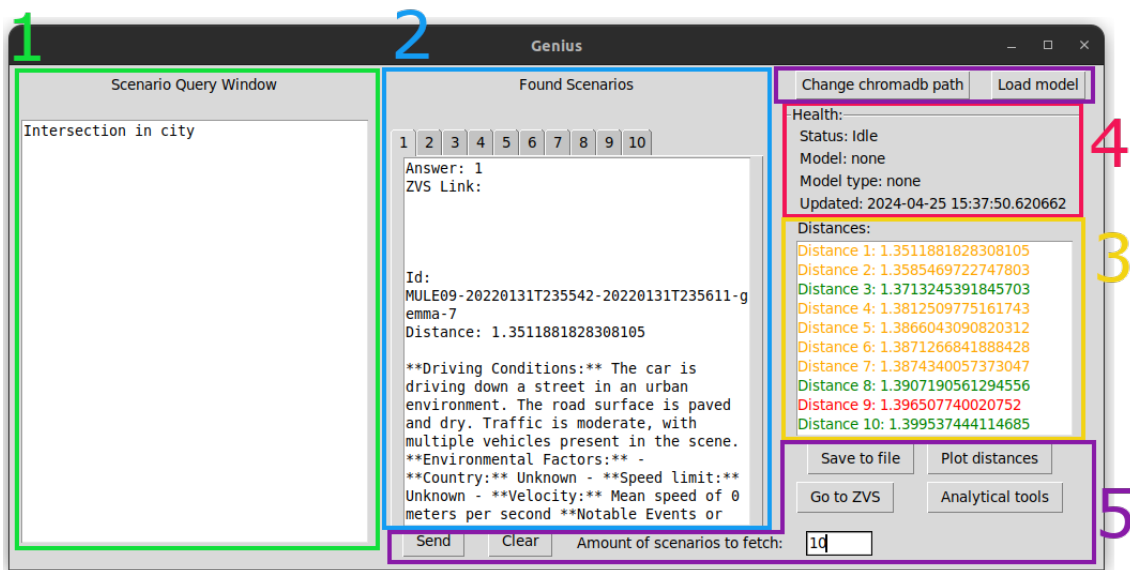


Figure 4.2: User interface for scenario querying.

Section 1: Scenario Query Window

This section enables users to input their query using natural language descriptions of scenarios.

Section 2: Found Scenarios Window

Here, extracted scenarios resulting from the query are displayed in tabs. Each tab presents the scenario description alongside relevant metadata such as timestamp, ID, distance, and a link to a visualization tool provided by the collaborating company. Scenarios are ordered by ascending distance, prioritizing closer matches. Scenario descriptions undergo keyword search highlighting to emphasize relevant terms.

Section 3: Distance List Window

This section lists the distances for each extracted scenario in ascending order. Distances are accompanied by color-coded labels generated through keyword searches: green denotes scenarios containing all query keywords, orange indicates some keywords, and red signifies none.

Section 4: Status Window

A status window provides users with real-time updates on the API's operational state and health. It displays whether the API is active or idle, along with details on the currently GPU loaded model type. Updates occur at one-second intervals.

Section 5: Miscellaneous Buttons

Utility buttons and an entry field enhance user experience. These include a 'Send' button for query submission (with 'Enter' key mapping for convenience), a 'Clear' button to reset the interface, and analytical tools for evaluating search results. Additionally, a 'Go to ZVS' button facilitates direct access to the selected scenario in the visualization tool provided by our collaborating company. Lastly, buttons for modifying the 'chromadb' path and loading new models support experimentation with different embeddings and models.

5

Findings

5.1 Cycle 0 Findings

The aim of this cycle was to determine what preliminary steps were needed to be performed in order to proceed with the primary purpose of this thesis.

The framework and API developed in this cycle will also be utilized in another master thesis at the collaborating company, therefore parts of the work in this cycle has been completed in collaboration with the author of that thesis.

5.1.1 Problem

In order to conduct the work in this thesis, we first had to set up an in-house LLM framework. To better understand the problem, we had meetings with engineers proficient in the server infrastructure at the company, allowing us to not only get a better grasp of what we needed to do but also to request the hardware specifications needed for the thesis. Furthermore, the meetings provided us with a grasp of the possible hardware limitations that could prove an issue to the solution.

5.1.2 Solution Candidates

From the problem investigation conducted in the previous step, two options were considered. The first option is to implement the in-house LLM solution on a local virtual machine (VM), that we could communicate with by creating a simple API framework. This option involves downloading the models to the virtual machine from Hugging Face and then deploying them to run on the GPU in the VM. To create the API to communicate with the models, the FastAPI framework in Python would be used.

The second option was using OpenShift, a cloud-based kubernetes platform. This would allow us to install the LLM framework and models on the OpenShift platform and allowing us to run API calls from our computers. The benefits of this solution in regards to the VM solution would be that the hardware resources could be expanded easily on demand and could be shared with other teams working simultaneously at the collaborating company.

5.1.3 Validation

The two solution candidates were further discussed with our supervisors at the collaborating company to get insight in how the possible solutions could be implemented using their infrastructure.

Using OpenShift for the thesis would require a process of requesting the specific hardware resources and having them approved, which would cost us time. If the final artifact were to be deployed with many users, deploying models to OpenShift would be the better alternative due to the ability to scale the resources. However, since the artifact was developed for the purposes of this thesis, these factors are not as important.

If we instead deploy the models on a VM, work can start immediately instead of waiting for approval. Using a VM instead of OpenShift would limit us to less powerful hardware resources and with less ability to scale up if we need more powerful hardware later on. We would also need to implement our own API to be able to access it, compared to OpenShift, where an API is already integrated in the service.

Based on these factors and our previous experience with VMs we decided that using a VM was the best solution. The main factor behind the decision was the time to when we could start developing and the ease-of-use of a VM compared to OpenShift. If the limited hardware resources of the VM would show to not suffice later in the thesis, a switch to OpenShift could be considered.

5.1.4 Implementation

To implement the LLM solution in-house, first we created a VM instance with the help of engineers from the collaborating company, on which the LLM models and framework along with an API function were implemented. At this point in order to use the LLM functionality, the API has to be running on the VM in order to call the API. This implementation allows both members of this thesis and the collaborating thesis student to use the LLM solution independently of each other.

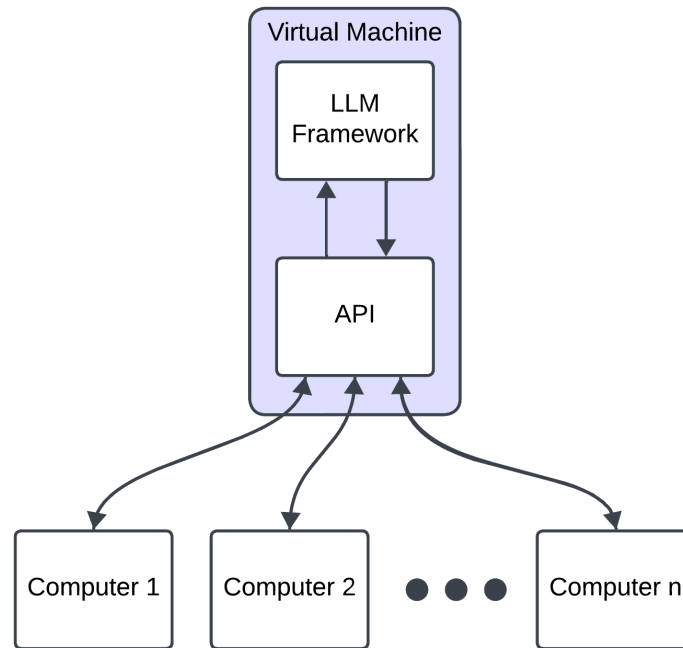


Figure 5.1: Flowchart for the in-house LLM solution

5.1.5 Evaluation

The first requirement the in-house solution should meet is that it should allow us to utilize an LLM model hosted locally to answer our queries, it should also allow us to embed text and store it in a vector database from which we can then semantically search, a metric we named features.

Features

The available features the solution has capability for.

In this aspect the solution shows great results, allowing us to query LLMs both textually and with images, returning generated answers. The solution also features embedding to and searching semantically from a local vector database.

Furthermore the solution was evaluated to the following additional metrics:

Speed:

The duration from when we make an API call to the VM to the moment an inferred answer is returned.

In this metric we don't aim to reach industry standard inference speeds as we are working with limited hardware. Instead we aim to reach inference speeds that doesn't impair our work severely. To test the call speed, we queried Mistral 7b quantized to a 4-bit resolution 20 times and measured the time it took from querying to the response being returned from the API. Using this method we measured a mean call speed of 8.24 seconds as shown in table 5.1, while this does not compare well to the speed of other industry standard LLM applications such as ChatGPT, it is an acceptable speed for our purpose. As the inference will only occur during processing of the table, the speed is not as vital as for a chatting application.

Mean call speed	8.24 seconds
Min call speed	2.81 seconds
Max call speed	12.35 seconds

Table 5.1: Measured call speed using a quantized Mistral 7b model with 20 samples.

The embedding search is much faster and, measured at a sample size of 5, shows a mean speed of 0.915 seconds, which is an acceptable speed.

Versatility:

The ease of which different models and capabilities can be interchanged.

The solution allows us to install several models simultaneously on the VM and to switch between them quickly. By loading the LLM models on the GPU and the embedding model on the CPU it allows us to use both simultaneously.

Simultaneous User Capacity:

The capability of allowing several users to use it simultaneously.

By using a VM and API solution, it allows all users of the solution to use it independently on their own computers and to easily change the front-end to fit their own needs. Due to the fact that two inferring instances cannot proceed simultaneously a 'status window' and queue system was implemented where a user can easily see what model is currently loaded and if it is in use, and allowing one to queue their queries.

5.1.6 Cycle 0 Learnings

From this cycle, we gained knowledge on how to implement and utilize an in-house LLM solution to leverage LLM models for our purpose. We realized that due to hardware constraints, we would have to limit the thesis to small LLM models, no larger than 7 b parameter models, additionally these will need to be quantized to 4-bit resolution due to the limited VRAM. This will still allow us to leverage the LLMs to produce good responses at an acceptable speed which will allow us to

conduct a proof of concept for our solution.

5.2 Cycle 1 Findings

The primary purpose of this cycle is to evaluate the problem and possible solutions, followed by a small-scale solution. This should allow us to answer **RQ1** and **RQ2**. A secondary goal was to test and evaluate the best LLM model for our purpose.

5.2.1 Problem

To better understand the vehicle logs and the current state of the searching functionality at the collaborating company, we first had a discussion with an engineer at the company familiar with the current system. This allowed for clarification of issues inherent in the usage of the current system as well as some of the intricacies of the vehicle logs, such as how some signals worked, which signals were most important to consider, and what resources were available that could potentially be of help. Furthermore, access to vehicle logs and camera data allowed us to get familiar with the data.

Additionally, to achieve optimal results for Genius, we need to use the best LLM model that we can get access too. Therefore, an evaluation of available models has to be conducted in order to determine the best model for our use case.

5.2.2 Solution Candidates

During our evaluation of the problem, we realized that there were several different methods that could lead to viable solutions. Two such methods were further analyzed and outlines were created for both candidates.

Solution 1:

The first of the considered solutions has an LLM in the loop. The following steps outline the plan:

1. **Task decomposition:** The initial query from the user would be processed into sub-tasks to handle. Each sub task would go through the following sub-steps.
 - (a) **Find relevant signals:** Key words would be extracted from the original query and embedded to find semantically relevant signals. Potentially an LLM would be prompted to provide more suitable key words and phrases.
 - (b) **Ask refinement LLM if the signals provided are sufficient for an answer:** An LLM would be queried if more signals from a given list are needed to provide an accurate answer; if so, the LLM would generate

a new query based on the original query and the LLM input on needed data and a new vector search would be conducted.

- (c) **Process data from relevant signals:** A processing script would run on the signal data from the extracted signals aiming to extract descriptive qualities about the data, such as type of data, timestamps for interesting points in data such as global as well as local minima/maxima, percentage of boolean states (if boolean signal) etc.
 - (d) **Prompt LLM with data:** The LLM would be prompted to give an answer to the original sub-task with a minimized data sequence from the signals, the data generated from the tools step as well as the signals.
2. **Join answers:** Finally an LLM would be prompted with all the sub-task answers and the original query to give a summarized answer to the user.

Solution 2:

The second method uses the LLM solution as processing tool instead. This method involves pre-processing entire vehicle logs to generate scenario descriptions. Each scenario descriptions would attempt to describe the current situation the car is in for the given time-span. By taking a few key-signals from the vehicle logs that are vital to understanding the scenario that the car is in along with an image from the cars camera system, the LLM generates a description using the given relevant data. These descriptions are linked by timestamp to the table and when a user queries Genius for a given scenario, the query is embedded and the descriptions that are closest semantically are returned along with a link to the table. The outline of the plan is as follows.

1. **Dividing the vehicle logs into scenarios:**
The first step is to divide the entire log into smaller pieces, each piece should correspond to one 30 second scenario.
2. **Processing of the key signals:**
A selection of signals are processed using simple scripts to condense important information to a shorter text format.
3. **Generation of description:**
A multimodal model is queried to generate description based on the camera footage. A second LLM is queried to combine this description with the data extracted from the signals, resulting in a description of the scenario. This description is embedded and stored in a vector space within the database.
4. **Semantic search for relevant scenarios:**
When the processing step is complete, a user is able to query the database in natural language and the semantically closest scenarios are returned using from the database of vectors as pictured in Figure 5.2.

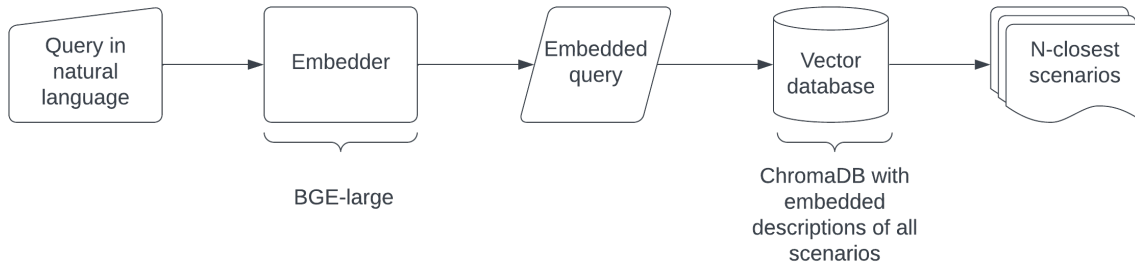


Figure 5.2: Querying for a scenario with Genius.

LLM model evaluation:

In order to identify the most suitable model for our specific use case, we decided to first search and compare available models using the Hugging Face leaderboard [35]. This approach enables us to narrow the selection down to four models that would warrant further testing and evaluation to determine the optimal model for the thesis. Our evaluation strategy encompasses both quantitative and qualitative assessments

For this evaluation we want a set of scenarios that we can test on a smaller scale using several model interchangeably.

Small-scale solution:

For this small-scale solution, a set of scenarios was first selected. This set consists of 8 scenarios of 4 types; each type distinct from the others to increase the semantic distances between the scenarios and allow easier testing and searching while also testing the dynamics between similar scenarios. The scenarios consisted of two scenarios of cars driving in snow, in a tunnel, by or on an off-ramp and during a speed limit change. The scenarios used were taken from Zenseact test drives.

A second set of scenarios was created by processing and embedding each scenario for 10 iterations, totaling 80 embedded scenarios, to emulate a larger set in a controlled manner.

Quantitatively, we devised a plan to query the second test set of scenarios with a set of queries for which we already know the correct answers. The distance between the correct answers against the incorrect answers would be plotted and evaluated with a selection of metrics to determine the model that performs best quantitatively. The metric scores are derived from the distances the scenarios each have to the query. The following are the quantitative metrics:

- **Mean Distance to Correct Answers**
Measures the average distance to correct answers
- **Mean Distance to Incorrect Answers**
Measures the average distance to correct answers
- **Mean Distance Difference**
Measures the relative distance between an average correct answer to an average

incorrect answer

- **Highest Distance to Correct Answer**
Measures the furthest correct answer from the query
- **Lowest Distance to Incorrect Answer**
Measures the closest incorrect answer to the query
- **Smallest Distance Difference**
Measures the smallest difference between correct and incorrect answers
- **Average Standard Deviation of Scenarios**
Measures the spread of the scenario distances

Qualitatively, evaluation is done by generating multiple descriptions for diverse scenarios using each model and carefully reviewing and cross-examining these outputs manually. We would subjectively assess the models against specific metrics and using these evaluations to inform our qualitative assessment of the models. The following are the qualitative metrics:

- **Trustworthiness** *Assess the accuracy of the information in the description produced in relation to the input scenario data.*
- **Comprehensiveness** *Measures how effectively the descriptions produced are in addressing the provided prompt.*
- **Hallucinations** *Refers to situations when the model generates inaccurate or illogical descriptions.*

5.2.3 Validation

During the solution determination phase of the thesis, regular meetings were held with our supervisor at the collaborating company. In these discussions, both proposed solutions were thoroughly examined in terms of feasibility, estimated time to implement, and how well they would potentially solve the issues experienced at the company.

While Solution 1 was acknowledged as a feasible approach and if successful could be a possible solution to some currently experienced problems with the current solution, it would introduce a prolonged user wait time and limited exploration avenues once the primary objective is achieved.

Conversely, Solution 2 exhibited comparable viability and functionality to Solution 1 but would have significantly reduced user wait times. This is due to the fact that while solution 1 depends on several LLM calls after a user enters their query, in solution 2, these calls are already done in a pre-processing step and when the user enters a query there are no calls to an LLM. Additionally, Solution 2 presented intriguing research opportunities that could be pursued beyond the scope of this

thesis, for instance, the resulting descriptions could have possible uses as label data for training a custom model in the future. Consequently, Solution 2 was selected as the solution for further development.

The Open LLM Hugging Face Leaderboard [35] serves as an established benchmark within the industry for comparing LLM models, thereby inherently validating the initial phase of our model evaluation. Additionally, we conducted a meeting with our academic supervisor to deliberate on the subsequent phase: the final custom evaluation. This collaboration ensured a solid foundation for our evaluation.

5.2.4 Implementation

5.2.4.1 Artifact Implementation

The implementation of the solution was completed in the following steps:

1. Dividing the vehicle logs into scenarios:

In this initial step the vehicle logs are divided into smaller parts. Partly so that it will be easier to process and won't pass the token limit for the LLM model but also to capture dynamics on a shorter timescale. The larger the time span of a car drive that is looked at the more homogeneous the description would be to other scenarios.

While the benefits of shorter scenarios are that more dynamics can be captured and more data can be processed, there are also some drawbacks. One drawback is processing time; having shorter scenarios has the same impact as having more scenarios to process and due to the fact that each LLM query takes approximately the same time to infer regardless of input length, this would equate to longer processing times the shorter we divide the scenarios. Another drawback is that some dynamics might be lost at too small a time-scale, for instance if one wants to capture a red traffic light signal with a following brake pattern, this would be hard to capture on the time-scale of one second.

Considering these factors we decided to divide the vehicle logs into scenarios of 30 seconds duration. This will allow us to go into detail on the data to process while capturing both minutiae and larger dynamics.

2. Processing of the key signals:

In order to extract interesting information from the ~6000 signals in the vehicle logs, we decided to extract a few key signals. Using all or even most signals would be impossible due to token limit constraints and probably ineffective due to flooding the LLM with irrelevant information, instead, by extracting only a subset of signals, the LLM can focus on the important information. We asked engineers at the collaborating company who have experience in querying the logs which signals are the most important and common for them. These signals along with some signals with general information about the drive were added to processing system.

These signals are then subjected to a processing phase. Each signal is processed based on how the signal works. For instance binary signals, such as a signal that observes whether the car is in a tunnel or not is simply processed to if is active or not during the scenario and for how long. Most binary signals are also discarded if they

haven't been activated during the scenario so to not clog the LLM with irrelevant information. Other signals may be subject to more advanced processing such as the velocity signals which is processed to return, mean velocity, the general dynamic of the curve and max and min velocity.

The output of the signal processing is then saved in a list to be used in the following step.

3. Generation of the description:

The description of the scenario is based on a combination of the signal data and the camera footage. To get usable information from the camera footage, it is prompted to a multimodal model. Tasked to generate a description of the scenario based on the footage, the multimodal LLM returns a descriptive text. A second LLM model, more capable at text generation, takes the camera description along with the processed signal data in the form of a short text and generates a new description that takes both signal data and the camera footage into account. This description is then embedded and stored in a vector database. Each scenario description is stored along with metadata that links them to the correct place in the vehicle logs by use of their timestamps. This concludes the processing phase of the solution.

4. Semantic search for relevant scenarios:

When a user want to use the solution to search for specific scenario within the vehicle logs, they simply enter a query in natural language describing the desired scenario. This query is then embedded and by searching for the closest embedding in the vector database, the most similar scenario descriptions to the query are returned. The description along with the timestamp to link it to the vehicle logs are returned to the user.

Interface:

In order to make this solution easier to use, an interface was created. In order to use the solution efficiently both as a user and a debugger the interface was equipped with the following features:

Scenario querying:

The primary feature of the interface. Scenario querying entails the process of writing a query in a text window. This query gets embedded and compared to the embedded scenarios in the database, the closest scenarios are returned and the selected number of scenarios are shown by order of increasing distance. The scenario descriptions of the fetched scenarios can be viewed along some metadata, additionally a list of distances for each fetched scenario is shown as well for analytical purposes.

Fast switching between vector space databases:

In order to quickly switch between different Chromadb databases, which could be helpful when comparing different methods or models, a button that changes the Chromadb path was created.

Fast loading of models:

In order to load, or change models a button for that purpose was created, especially useful during the model evaluation process.

Small array of analytical tools:

To alleviate the evaluation process a number of analytical tools were created. These include a plotter, which plots the fetched scenarios to their distances, and some mathematical calculations that can be automatically conducted, measuring scores for each quantitative metric.

Status window:

To assist the user a status window was created. This windows fetches information from the API each second, displaying whether the API is currently in use and which model is currently loaded. This is especially helpful to avoid several users trying to use the API simultaneously with different models.

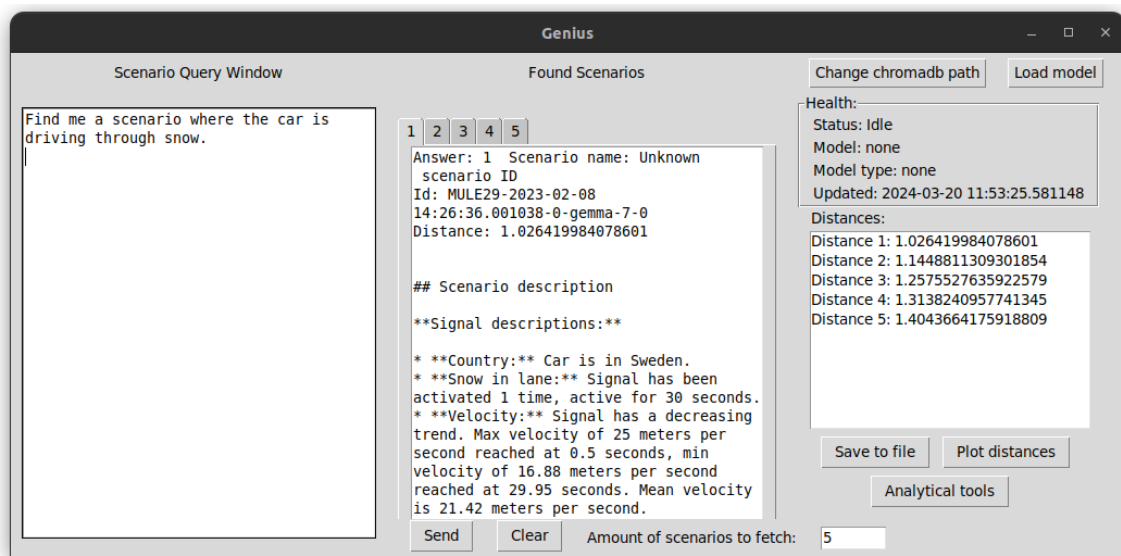


Figure 5.3: The interface that allows the vector database to be queried, currently queried for a snowy scenario.

5.2.4.2 Implementation of the LLM model evaluation

In order to evaluate the different models, both a quantitative and a qualitative evaluation was performed.

Quantitative model evaluation

To evaluate the models quantitatively, the second test set was used, containing a description of each scenario for 10 iterations, totaling 80 embedded scenarios. For each iteration the process has different results which allows for testing the variability

5. Findings

and stability of the process. This is done for each model to evaluate each in a similar way.

All models are quantized to a 4-bit resolution excluding Gemma 2b, which due to its small size, is only quantized to an 8-bit resolution.

To evaluate the effect each model has on the search functionality, the following simple test was constructed: Genius is queried for a scenario that is known to exist only in a set of the scenarios, the distances for each answer are then grouped to the scenario from which they originate and plotted to show both variation of distances for the same scenario and the difference of distances between correct and incorrect answers.

In the evaluation Genius is queried to find a scenario where the car is driving through snow, only two such situations exist in the test set, aptly named 'Snow 1' and 'Snow 2'.

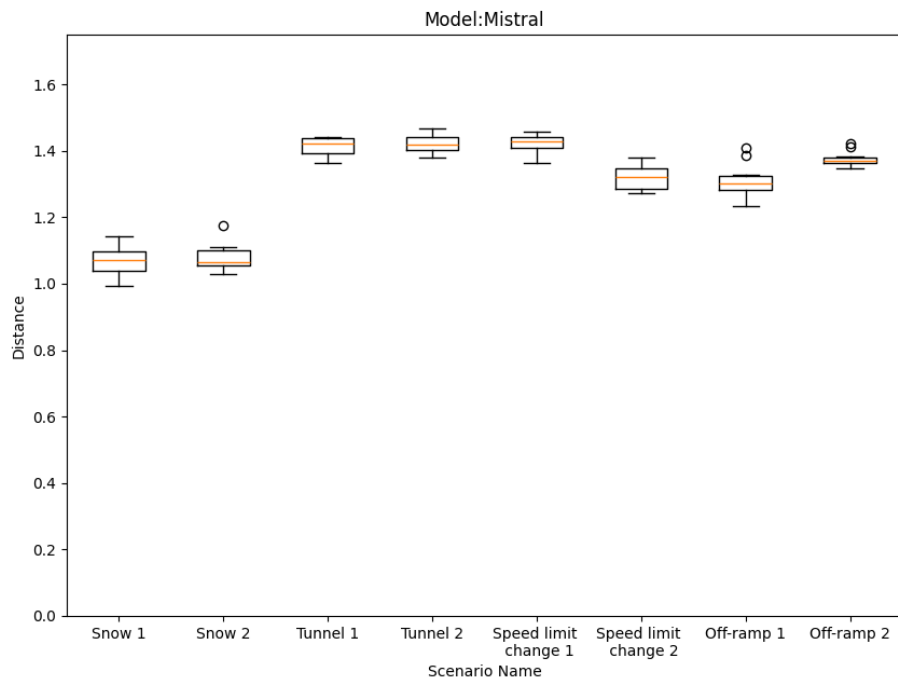


Figure 5.4: Histogram of the ratings of the generated scenarios.

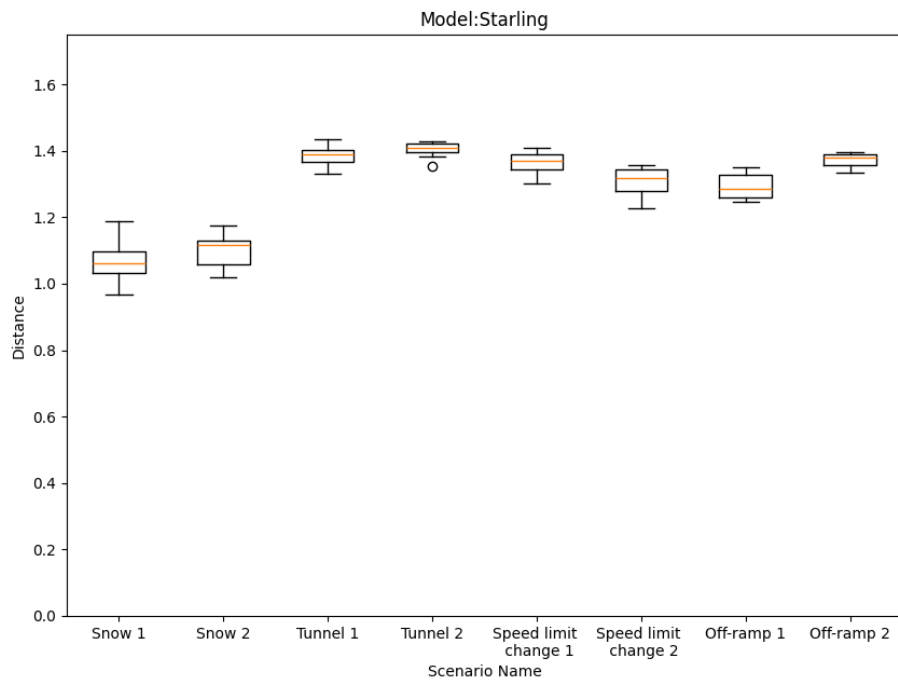


Figure 5.5: Histogram of the ratings of the generated scenarios.

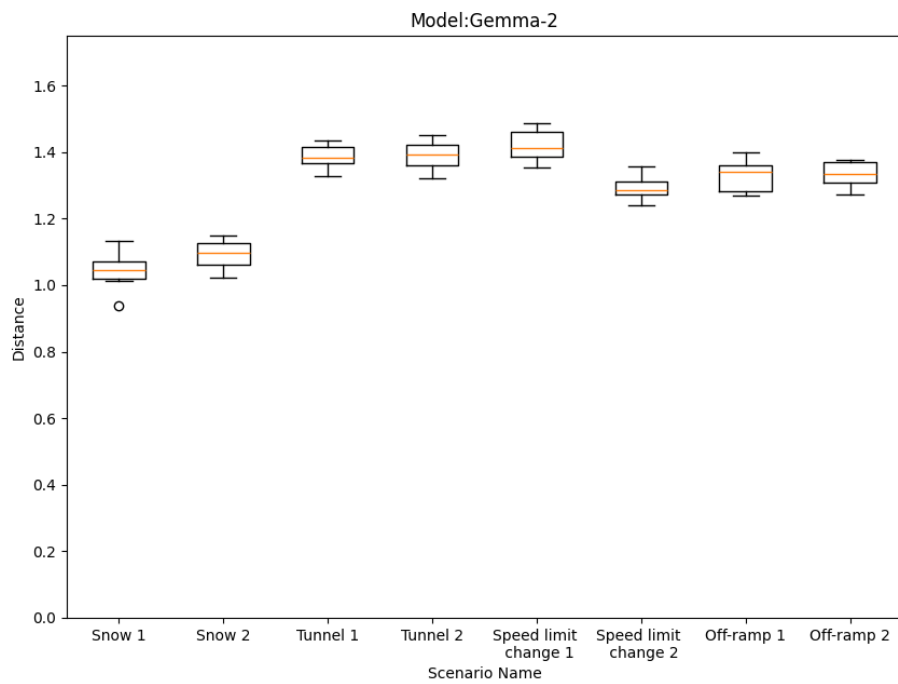


Figure 5.6: Histogram of the ratings of the generated scenarios.

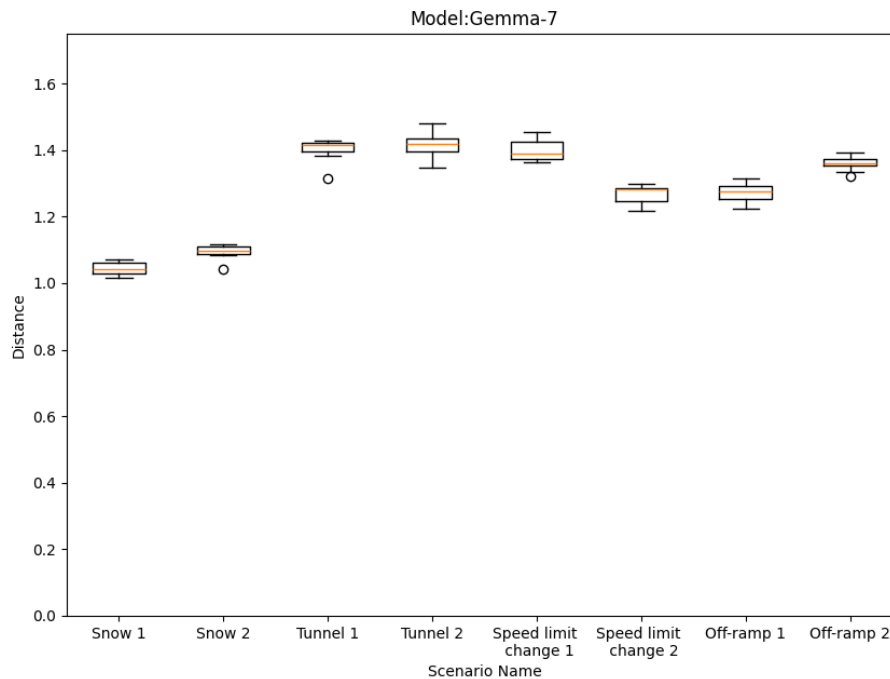


Figure 5.7: Histogram of the ratings of the generated scenarios.

	Mistral 7b	Gemma 2b	Gemma 7b	Starling 7b
Mean Distance to Correct Answers	1.093	<u>1.061</u>	1.081	1.083
Mean Distance to Incorrect Answers	<u>1.433</u>	1.411	1.419	1.417
Mean Distance Difference	0.340	<u>0.350</u>	0.338	0.334
Highest Distance to Correct Answer	1.156 (1.147)	1.137	<u>1.133</u>	1.202(1.139)
Lowest Distance to Incorrect Answer	<u>1.303</u>	1.289	1.261 (1.309)	1.289
Smallest Distance Difference	0.147 (0.156)	<u>0.152</u>	0.128 (0.176)	0.087(0.150)
Average Standard Deviation of Scenarios	0.033	0.041	<u>0.028</u>	0.038

Table 5.2: Table of comparison metrics for each model. The best performing metric for each category is underlined. Metrics when outliers are excluded are enclosed in parentheses.

Table 5.2 shows the comparison metrics used to evaluate the models. Emphasis should be placed upon the metrics 'Smallest Distance Difference' as well as 'Average Standard Deviation of Scenarios', as they show the distinction between correct versus incorrect answers and the stability of the answers respectively. Metrics where both outliers are considered and excluded are shown.

Initially, Gemma 2b shows promising results, being the model with the most outperforming metrics. However it shows the worst stability shown in the 'Average Standard Deviation of Scenarios' metric.

When outliers are excluded the results take a different shape, now Gemma 7b shows by far the best results, outperforming the others in most metrics, most importantly in both 'Smallest Distance Difference' and 'Average Standard Deviation of Scenarios'.

Qualitative model evaluation

The qualitative evaluation of the models is based on the following criteria.

- **Trustworthiness**

Assess the accuracy of the information in the description produced in relation to the input scenario data.

The accuracy and inclusion of all relevant signals is a vital part of the description. In this aspect Gemma 2b showed inauspicious results, often leaving out important information, failing to mention all relevant signals. For instance, when describing scenarios that include an upcoming speed limit change, it has failed to mention it every single time. Fortunately other models show more promising results, especially Gemma 7b has mentioned all relevant signals without fail during the evaluation process.

- **Comprehensiveness**

Measures how effectively the descriptions produced are in addressing the provided prompt.

The ability to follow the directions given in the prompt and produce structurally homogeneous descriptions is important to design the pre- and post-processing of the descriptions. In this metric most models struggled, Gemma 2b was prone to writing short paragraphs with little information, often with a foreword unrelated to the description. Mistral 7b tended to generate well-written text but overly long, often 3-4 paragraphs. Starling 7b and Gemma 7b performed well in this category as both produced concise but informative descriptions without superfluous text, although Starling 7b was prone to write in a more flowery, poetic way which could clash with how engineers would phrase their queries.

- **Hallucinations**

Refers to situations when the model generates inaccurate or illogical descriptions.

LLMs have a tendency to hallucinate answers when the model has insufficient knowledge to generate an answer. In this aspect, Starling 7b stood out as the model that most often generated descriptions that were more creative than factually correct. Examples include sentences like *"The car was initially speeding, but it is now back within the speed limit, as shown in the image of a car driving down a quiet street with a clear blue sky."*, which is a logically unsound statement, one cannot deduce that the car was speeding from an image of the car driving in good weather. Other models also produce hallucinations, such as Mistral 7b: *"The car suddenly brakes for a short duration due to a momentary distraction, causing the driver's eyes to leave the road for 2.025 seconds."* In this example, Mistral 7b combines two signals, one showing the total time of the drivers eyes of the road and one showing the total brake time, and deduces

that they are correlated and that a distraction had occurred. This could be correct but there is no way of knowing and the model should not make such claims.

5.2.5 Evaluation

The purpose of this cycle was to implement a small-scale solution. The evaluation will be performed partly with assistance from engineers at the collaborating company and partly mathematically by analyzing the answers and their distances to the query and other answers. In the first step, the model evaluation will be performed so that the optimal model can be focused on in the small-scale solution evaluation.

5.2.5.1 LLM model evaluation

Quantitative model evaluation

In 5.4-5.7 the distances to each scenario are plotted in box plots for each model. Here, the compactness of the box plots shows less variability in answers, and therefore a stability that is sought after, and the distance between the correct and incorrect answers show the distinction of distance between related and unrelated scenarios. The higher the distances between the correct and incorrect scenarios, the easier the search function will have to find the correct answers and eliminate the incorrect answers from the result.

Analyzing the Figures 5.4-5.7, Gemma 7b displays the most compact distance variation and a good separation of correct versus incorrect answers.

Qualitative model evaluation

To evaluate the models, the descriptions for several scenarios were examined and evaluated according to the above metrics. To showcase the models' performance against each other, one scenario was selected, and each of the models generated description of the scenario was cross-examined. In this scenario, the most interesting event was that an upcoming speed limit change was present. The full descriptions of each model can be read in the Appendix A.1.

Each model is ranked between 1 and 3 for each metric, where 3 is considered good and without any major recurring mistakes, 2 is sub-par with recurring mistakes and 1 is considered unacceptable, with mistakes occurring a majority of the time.

	Mistral 7b	Gemma 2b	Gemma 7b	Starling 7b
Trustworthiness	2	1	3	2
Comprehensiveness	2	2	3	3
Hallucinations	2	2	3	1

Table 5.3: Table of different qualitative comparison metrics for each model.

Based on the results of both qualitative and quantitative metrics, the model that best suits the problem is Gemma 7b. Therefore in the continuation of the project this is the model that will be used.

5.2.5.2 Artifact Evaluation

Figure 5.8 displays a simplified visualization of the semantic space in which the query and scenario descriptions exist visualized with t-SNE. Here, Genius was queried to find scenarios where the car was driving through snow. The scenario set used was the same set used for model evaluation to emulate a larger set. Analyzing the picture, it is clear that all correct scenarios are clustered close to the query in the semantic space, while all other scenarios are found much further from the query. Additionally it is positive, and not surprising, that all descriptions that stem from the same scenario are clustered together showing stability in the description and embedding process.

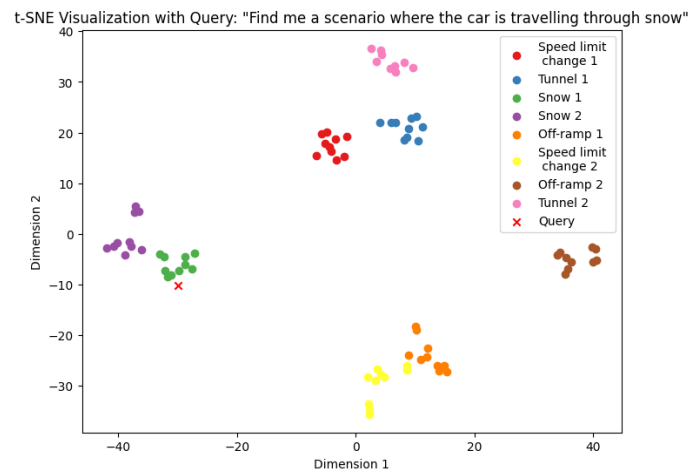


Figure 5.8: t-SNE visualization of the test scenario set descriptions when queried for a scenario containing snow.

Figure 5.9 shows a histogram of each of the 8 scenario descriptions distances between each other, plotted to show the distribution of the scenarios in the semantic space. An ideal distribution for the thesis purposes would be uniform, as that would mean that the scenarios are distinct and not clustered together. However in this small-scale solution the distribution is not completely uniform. This is likely due to two circumstances: firstly, the small data set will show irregularities greatly, with a larger data set the distribution would more likely converge to a uniform distribution. Secondly, in this small data set, there scenarios that share more similarities than others, for instance three of the scenarios include an off-ramp in their description, this leads to those scenarios being closer to each other off-setting the distribution.

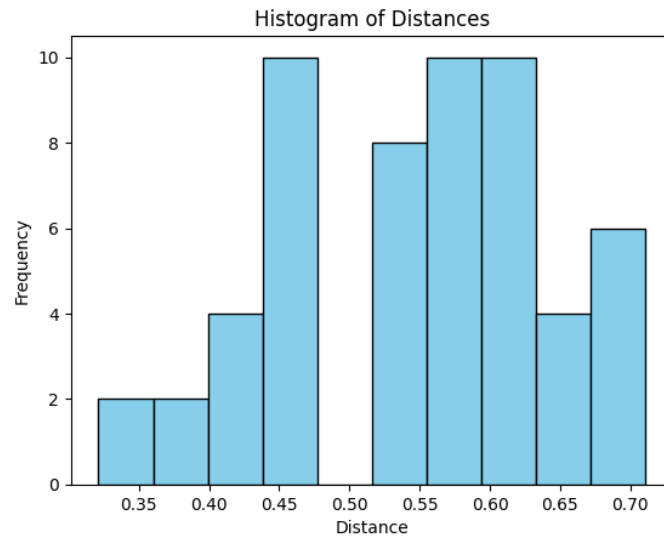


Figure 5.9: Histogram of the distances between each of the 8 scenario descriptions generated by Gemma 7b

Scenario description evaluation

To evaluate the quality of the scenario descriptions generated, a form was sent out to engineers working with the log data at the collaborating company. The 8 scenarios is split into 4 forms with 2 scenarios in each. The forms includes links to an internal visualization tool for the specific scenarios along with the corresponding descriptions generated by Genius. The developers were asked to grade the description and point out any areas of improvement that could be worked on along with any factual errors in the description. The entire form can be found in Appendix A.3.

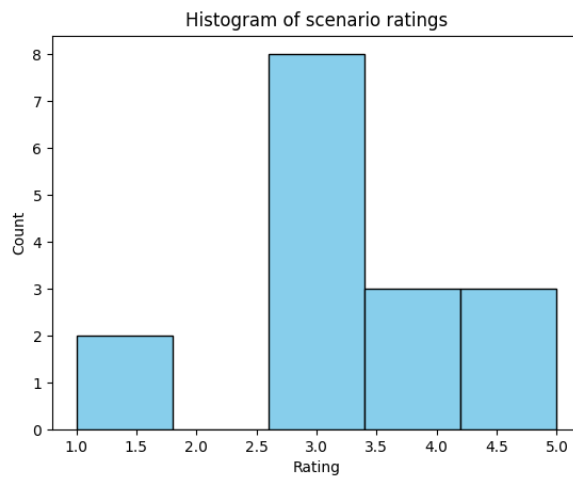


Figure 5.10: Histogram of the ratings of the generated scenarios.

Rating	Description
1 star	Poor description, several important aspects missing.
2 star	
3 star	Acceptable description of the scenario, capturing some of the important aspects.
4 star	
5 star	Great description, all important aspects of the scenario is captured in the description.

Table 5.4: Description of the rating scale in the form.

The 16 responses resulted in a mean value of 3.3125. Which according to our scale of the ratings in Table 5.4 is "Acceptable description of the scenario, capturing some of the important aspects.". Out of the eight scenarios there was only one scenario that had a score under 3, the same scenario was rated with twice with the score 1.

Number	Comment
1	"There are other vehicles. "
2	"There are quite many cars on the road. Image description and overall scenario are both saying no other vehicles"
3	"There are other vehicles present. I think the car is in the rightmost lane, the yellow area is not a lane."
4	"Image description only give the description of the beginning of the scenario."
5	"There are other vehicle present in the video."
6	"It should be an urban area, not a rural or suburban area. It's failed to detect the vehicles in the scenario, it has vehicles presented all the time."
7	"There are other vehicles present in the video. This isn't a highway. It isn't dark. The road isn't particularly straight"
8	"There is a curve during the drive. Failed to detect vehicles in the scenario. Driving in the daytime not in the dark. Road construction sign is missin"
9	"Rainy day "
10	"No speed limit change sign Temporary parking place is missing "
11	"there are other vehicles in to opposite lane "

Table 5.5: Comments on Question 2/6: "Are there any factual errors in the description of the scenario?".

From the comments in Table 5.6, the description often seems to miss that there are other vehicles in the scenario, often stating that the road is clear when it is not. This could be a result of the limited multi-modal model resolution being limited (336x336 pixels) and that it has difficulty distinguishing vehicles in the image. There was also a comment about that the image description is only from the beginning of the scenario, which is expected since we only utilize the first frame of each video in the processing.

Number	Comment
1	"The car seems to be holding out for a vehicle that overtakes it on the right side in one of the intersections."
2	"There are quite many cars on the road. Image description and overall scenario are both saying no other vehicles"
3	"There is a road work (traffic cones) present in the beginning of log, would be useful to know."
4	"speed limit change/speed limit sign"

Table 5.6: Comments on Question 3/7: "Is there anything important missing from the description? For instance, any specific signals missing that you think would improve the description."

The description is missing some information that should be available in the signal data that is processed, namely whether there is a speed limit change or roadwork in the scenario. Looking at the signal data, the scenario where road work was not detected, the signal for road work was active, so our model missed that in the description. However, the scenario where the speed limit change was missing, the signal for speed limit change was not activated during the scenario, so we could not have detected it from the signal data. The multi-modal model could have detected both of these from the video, but since we only use one frame, it would have needed to be in the first frame.

Number	Comment
1	"good catch with tunnel and off ramp."
2	"Description seems fail to detect vehicles on the road or parking along the road."
3	"Good catch with the off ramp, weather condition and country"
4	"Good catch with the upcoming speed limit change, the speed limit doesn't change during the video but I can see in the distance in the last frame that "
5	"The descriptive words like "indicating cold and potentially challenging driving conditions" are not needed, just describe the truth should be enough."

Table 5.7: Comments on Question 4/8: "Other comments."

Comment about the descriptions using unnecessary wording such as "indicating cold and potentially challenging driving conditions", this is something that we have tried

to limit through prompting but have not succeeded entirely. The quite heavy quantization of the model could also influence the models ability to follow our prompts.

5.2.6 Cycle 1 Learnings

In this cycle we implemented a smaller-scale solution to an acceptable degree of success. From the evaluation of the LLM models, Gemma 7b was shown to have the greatest overall performance and will be the model of choice in the artifact. Evaluation of the generated descriptions showed that they were of acceptable quality and the same scenario processing will be kept for the subsequent cycle.

The results of embedding the scenarios show promising results in terms of grouping similar scenarios and separating them from other scenarios. However, since it is at such a small scale, no real conclusion can be drawn regarding **RQ3** in this cycle. In the subsequent cycle, more data will be added to evaluate how well the semantic search works on a more representative scale.

5.3 Cycle 2 Findings

The primary purpose of this cycle is to scale the artifact of cycle 1 and evaluate the solution on a larger scale. The secondary purpose is to improve the current solution using what we learned from cycle 1.

5.3.1 Problem

In cycle 1, we evaluated the solution with a small set of scenarios, much smaller than the size of the actual data set for which our solution is aiming to be utilized, and therefore the solution needs to be tested and evaluated, and perhaps tweaked, on a much larger scale. When engineers search for logs using SQL, they utilize thousands of separate log files, each with upwards of 1000 scenarios each. If this solution were to be deployed at full scale, millions of scenarios would have to be processed to have access to the same breadth of information as existing tools.

In Cycle 1, since we were using just eight scenarios, they were downloaded and matched to their respective video files manually. Doing this for many more scenarios would take a long time; therefore, an automated scenario matching pipeline will have to be implemented.

Additionally, one problem with the current solution is that it can be unclear if the returned answers are correct or not. When querying for a scenario, the scenarios with the closest distance will be returned; however, when a scenario that is queried for does not exist, the solution will still return the closest answers, which would then be incorrect answers and the user would have to check the scenario to determine if it is correct or not. Therefore, in this cycle, a way to filter out correct answers from incorrect answers after the initial query and search will be implemented.

The descriptions generated by the artifact in cycle 1 did not always follow the same format; this not only makes reading the descriptions somewhat confusing but it

could also have an impact on the semantic embedding since it might take the format into account when generating the embedding vector.

5.3.2 Solution Candidates

In order to scale up the solution, a larger data set must be obtained. We propose the creation of a dataset comprising approximately 100 scenarios randomly sampled from diverse drives in various countries. We had access to a CSV file containing the file names of each of the video files; this filename included the name of the vehicle and the timestamp. This serves as a starting point for our proposed scenario matching pipeline.

The proposed scenario matching pipeline:

1. Create a list of all available log files containing signal data, these also include the name of the vehicle and timestamp.
2. Match the list of video filenames to the log filenames by extracting and comparing the timestamps.
3. Select 100 matches at random.
4. Save the rows corresponding to the video file for each of the log files.
5. Open and save the first frame of each of the video files.

A strategy was devised to distinguish correct from incorrect answers using Word2Vec. This plan involves extracting keywords from the query and subjecting them to the Word2Vec model to retrieve semantically closest words. Subsequently, these words are searched within the identified scenarios. The presence of similar or identical words for each keyword within a scenario enhances the likelihood of it being a correct answer, whereas the absence diminishes this likelihood. Consequently, scenarios where all keywords are found are categorized as green, those with only some keywords as orange, and those with none as red.

A secondary strategy to distinguish correct answers from incorrect answers that is derived from the distance values is planned to be implemented. Two candidates are proposed:

1: Z-Score

Z-score is a statistical tool used to measure the number of standard deviations a data point is from the mean. In order to use Z-score to find correct answers, a threshold Z-score is implemented and any data point with a Z-score less than the threshold is a potential correct answer. This can be shown by the following formula.

$$if s_{distance} < S_{mean} - t \cdot S_{stdev}, s \in C$$

Where S is the set of scenarios used, s is one specific scenario from the set, t is a constant threshold number, and C is the set of correct scenarios. This approach will essentially find outliers according to a parameter, t, and in the case of a large majority of incorrect answers the correct answers are likely to be outliers to the rest

of the set.

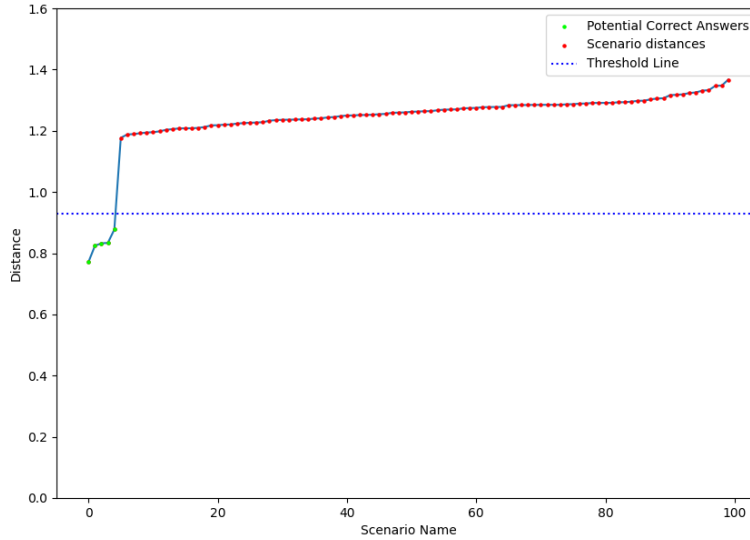


Figure 5.11: Example of a Z-score threshold fitted to a curve of scenario distances.

2: Linear Regression

Linear regression fits a line to a trend in a data set. By fitting a linear regression line to the ascending curve of the distances from each scenario to the query and looking at all answers below the first intersection of the line and the curve, we can find a small number of potentially correct answers. This can be shown by the following formula.

$$if\ s_{distance} < f(x_0),\ s \in C\ where\ min(f(x_0) = r(x_0))$$

Where S is the set of scenarios used, s is one specific scenario from the set, C is the set of correct scenarios, f(x) is the curve of scenario distances and r(x) is the linear regression line thus making x_0 the first intersection point.

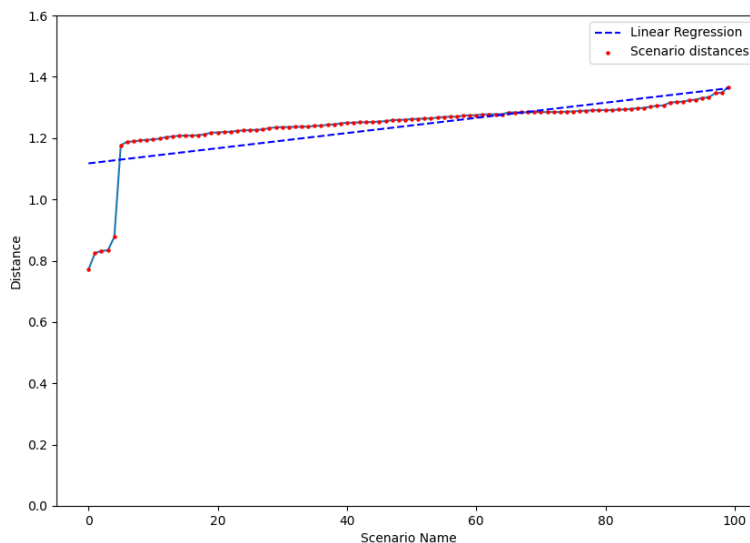


Figure 5.12: Example of a regression line fitted to a curve of scenario distances.

To improve the format of the generated descriptions, a possible solution is to use few-shot prompting, this would show the model how we would like it to answer and should make the model answer in a similar way.

5.3.3 Validation

The plan for scaling up the dataset was discussed with our supervisor at the collaborating company, where the size was deemed sufficient for our evaluation purposes and would serve as a closer approximation of real-world use. The pipeline for gathering and processing data for the scenarios was discussed and validated.

Similarly, the plan to distinguish between correct and incorrect answers using keyword search was also discussed with the same supervisor who believed it to be a valid solution to gain further validation of correct answers.

The validation of the secondary methods for identifying correct answers involved testing the Z-score and linear regression approaches on a small dataset to assess their efficacy. While both methods were evaluated, an issue emerged with linear regression: it consistently produced correct answers, even in scenarios where none existed, leading to false positives. In contrast, the Z-score method identified answers that significantly deviated from the norm, but struggled when a substantial portion of scenarios were correct, as these correct scenarios were not outliers. Despite the Z-score method exhibiting false negatives, its stability and ability to discern outliers made it the preferred choice for this thesis.

5.3.4 Implementation

Adding more scenarios first requires matching video files with log files that contain the signal information. To speed this up we implemented a script for automatically matching each video file with the respective log file and saves the corresponding file names in a list. This was achieved in a few steps, the first being to save lists of all the file-names for both video files and log files since these contain the vehicle number and both the start and end time. A list of video file names was already available and a list of all the log files was created using Python. To find the respective log file for each video file, the vehicle and timestamp in the file names was compared to find a log with the correct vehicle during the same time. The result of this step was a dataframe containing all the matching file names of videos and logs.

Each log file contains the signal information from the entire run, which could be a whole day, since each video file is just 30 seconds, we want to extract just the corresponding rows of the log file. First 100 random video files were sampled from the list of matched files, this was done to spread out what scenarios were present in the final database. Then the correct rows were found by iterating over the rows in the corresponding log file for each video file and matching the timestamp of each row to find where the scenario starts and ends. This was saved to a Pandas dataframe and saved as a compressed parquet file.

Currently we are just using the first frame of each video file, to extract it from each video we used OpenCV and loaded each video from the sampled list of video files and saved the first frame as a JPG file. Both the extracted rows from the log files and the frames from the videos were saved under the same name with different file endings to retain the information from the previous naming scheme, "vehicle-start-end" with either ".jpg" or ".parquet.zip" as ending. Finally all the scenarios were processed in the same way as in the previous cycle and saved to a new vector database.

To implement the keyword searching, a pretrained Word2Vec model, trained on Google News and stored as a binary file, was downloaded and stored in the VM. Subsequently, this model is loaded as an embedding model on the CPU. Following completion of initial querying and scenario extraction, the keywords of the query are entered into the model to retrieve similar words. This process generates a Python dictionary where each keyword serves as a key, and all similar words are listed as the corresponding values.

Next, the dictionary is iterated, and each keyword along with its similar words are sought for within each scenario description. When found, these words are visually emphasized by boldening in the text to enhance readability for the user. Subsequently, for each key in the dictionary, if a suitable word is detected within the text, the scenario is labeled as potentially correct, highlighted in green. Conversely, if only some words are found, indicating a lower likelihood of correctness, the scenario is marked as less likely correct, denoted by orange. Finally, scenarios where no relevant words are identified are labeled as probably incorrect, indicated by red.

It is crucial to note that the presence of all keywords in a scenario does not guarantee its correctness to the query. For instance, a query for 'speeding' might yield a

scenario describing a car 'not speeding,' containing the same keywords but conveying opposite meanings.

In addition to this, a regex expression was used to also add variations of the keyword by looking for the keyword along with suffixes, so for the keyword 'snow' words like 'snowy', 'snowing', 'snowfall' would also be looked for in the scenario description.

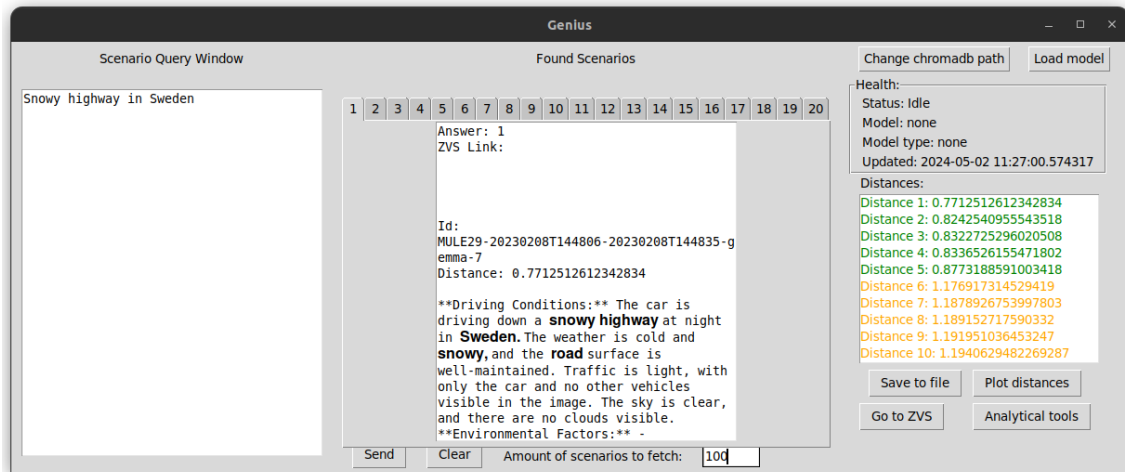


Figure 5.13: Image of the interface querying for a change of lane, showcasing the keyword search with emboldening and color labeling.

The Z-score method for validating correct answers was implemented as described in the solution candidates section. By applying the method on a set of queries with known answers, both correct and incorrect, while varying the threshold parameter, an appropriate value for the parameter could be found. A threshold of 3 standard deviations from the mean would find all answers when the scenario queried for is sufficiently different from the norm of the scenarios but would struggle to find more common, yet correct answers and it was found that while a threshold of 2 standard deviations from the mean, found almost all correct answers, it also selected incorrect answers from time to time, giving false positives. However, a threshold of 2.5 standard deviations from the mean, demonstrated the good of both worlds, boasting a capability to find most answers, even when the query is quite general, yet very rarely gives false positives. Therefore, going forward a threshold of 2.5 standard deviations from the mean will be used in this thesis.

Implementing the few-shot prompt is straightforward, all that has to be done is changing the prompt that is sent to the LLM and adding a few examples of the desired format. We created a few-shot prompt that includes two examples of previously generated descriptions. We made a few minor manual adjustments to the examples to simplify the structure of the text.

5.3.5 Evaluation

Quantitative evaluation

To evaluate the spread of the embedded scenarios we plot a histogram of the distance between each of the 100 scenarios, this was done in the same way as the evaluation in Cycle 1. Figure 5.14 shows that the histogram has a close to Gaussian form. Ideally, the distribution should be uniform since that would mean that the embeddings are evenly spread out in vector space. The reason the plot has a Gaussian form is likely due to the fact that most of the embeddings all contain information to driving scenarios, which is a limited domain of text.

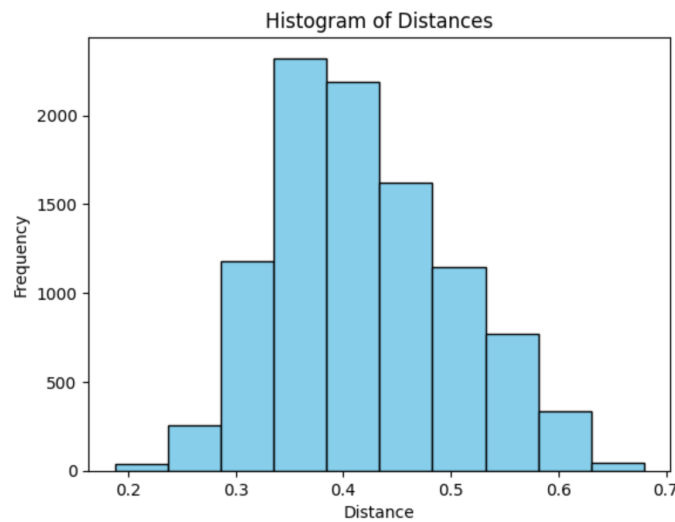


Figure 5.14: Histogram of the distances between each of the 100 generated scenario descriptions.

To quantitatively evaluate the solution for the 100 scenarios, a few varied queries will be analyzed using the following metrics:

- **Largest Gap**
Measures the largest gap in distance between adjacent scenarios
- **Min Distance**
Measures the distance to the closest scenario
- **Max Distance**
Measures the distance to the furthest scenario
- **Range**
Measures the range of distances, e.g. Max Distance - Min Distance
- **Standard Deviation**
Measures the standard deviation of the scenario distances
- **Relative Largest Gap**
Measures the Largest Gap relative to the Range

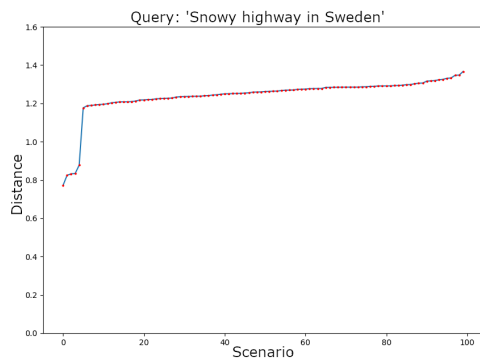


Figure 5.15: Plot representing the distances between a query and each of the embedded scenarios.

"Snowy highway in Sweden"	
Largest Gap	0.2996
Min Distance	0.7712
Max Distance	1.367
Range	0.5956
Standard Deviation	0.1031
Relative Largest Gap	50.3%

Figure 5.16: Table showing the measured results for the different distance metrics.

For the query used for Figure 5.15 and Table 5.16 there are two strictly correct answers and 3 additional close answers. The 3 close answers describe a scenario where the car is driving on a snowy road in Sweden but not a highway. This can be observed in the plot as there are 5 answers with distinctly lower distances, while the rest of the answers are shown as a slowly ascending line with fairly homogeneous distances. This is a fairly ideal scenario where there is a clear distinction between good answers and incorrect answers shown in the plot and the Largest Gap metric.

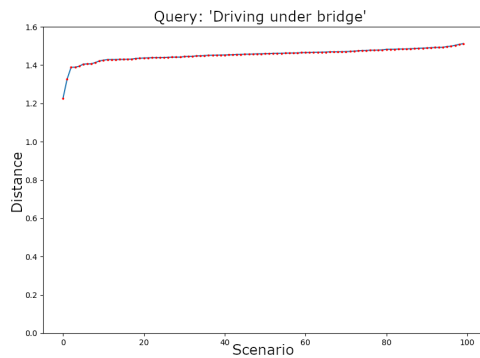


Figure 5.17: Plot representing the distances between a query and each of the embedded scenarios.

"Driving under bridge"	
Largest Gap	0.09962
Min Distance	1.226
Max Distance	1.512
Range	0.2858
Standard Deviation	0.03688
Relative Largest Gap	34.9%

Figure 5.18: Table showing the measured results for the different distance metrics.

Similarly the query used for Figure 5.17 and Table 5.18 exhibit two correct answers, while the rest of the scenarios does not contain a bridge in any form. Again this is observable in the plot as there are two distinct distances plotted followed by a slowly ascending line illustrating the incorrect scenarios.

5. Findings

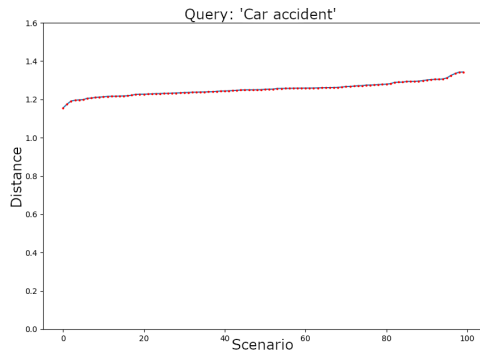


Figure 5.19: Plot representing the distances between a query and each of the embedded scenarios.

"Car accident"	
Largest Gap	0.0201
Min Distance	1.155
Max Distance	1.343
Range	0.1879
Standard Deviation	0.03477
Relative Largest Gap	10.7%

Figure 5.20: Table showing the measured results for the different distance metrics.

The query used for Figure 5.19 and Table 5.20 has 0 correct answers and was used to observe the results when querying for something that doesn't exist in the database. This can be observed in the curve of the plot, as it shows a slowly ascending line, similar to the ones following the correct answers in the previous plots. There is no clear distinction between any answers, as shown by the plot and the Largest Gap metric.

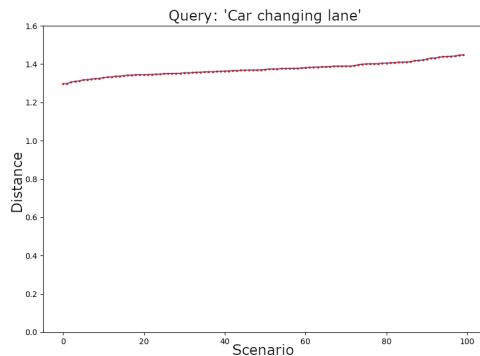


Figure 5.21: Plot representing the distances between a query and each of the embedded scenarios.

"Car changed lane"	
Largest Gap	0.009947
Min Distance	1.355
Max Distance	1.519
Range	0.1640
Standard Deviation	0.03494
Relative Largest Gap	6.1%

Figure 5.22: Table showing the measured results for the different distance metrics.

The query used for Figure 5.21 and Table 5.22 has 7 correct answers, however this is not clear from the figure. The 7 correct answers are among the 9 closest scenarios (two incorrect scenarios got a closer distance than one correct), but there isn't a clear observable distinction between correct and incorrect scenarios and in fact the curve is very similar to the previous one where no correct answers were available. This is likely due to the fact that the words 'lane' and 'change' are both used in most scenario descriptions in different contexts.

In order to gain an metric to evaluate correct from incorrect answers an Average

Relative Largest Gap (ARLG) metric is calculated. By querying a set of queries with known correct answers and taking the ARLG, a standard ARLG for correct answers can be calculated.

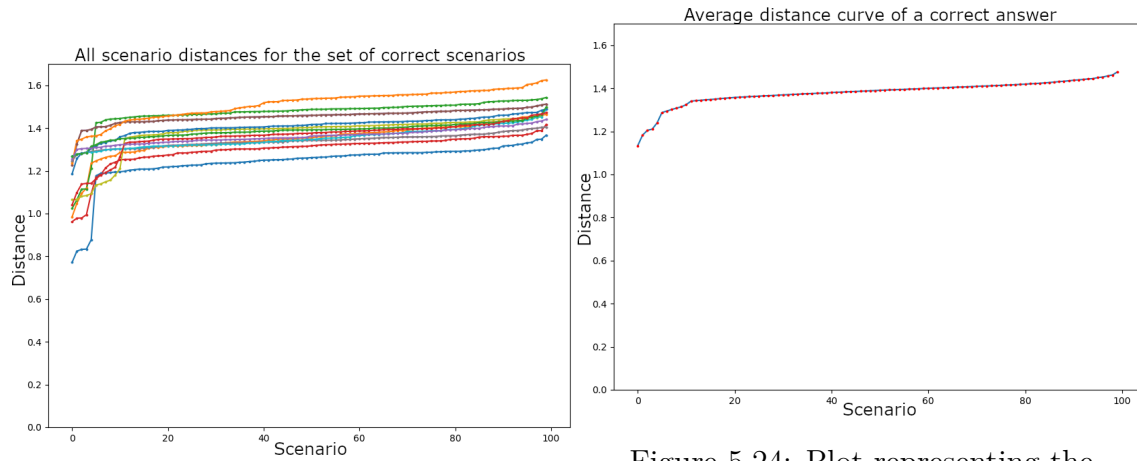


Figure 5.23: Plot illustrating all scenario distances for the set of correct scenarios.

Figure 5.24: Plot representing the distances between a query and the average distances of all scenarios in the set.

Average Correct Scenario Metrics	
Average Largest Gap	0.09903
Average Min Distance	1.113
Average Max Distance	1.476
Average Range	0.3421
Average Standard Deviation	0.05973
Average Relative Largest Gap	26.6%

Table 5.8: Table showing the average measured results for the different distance metrics measured from the set of correct scenario answers.

Similarly to find a standard for incorrect answers, a set of queries known to have no answers is queried and the ARLG is calculated.

5. Findings

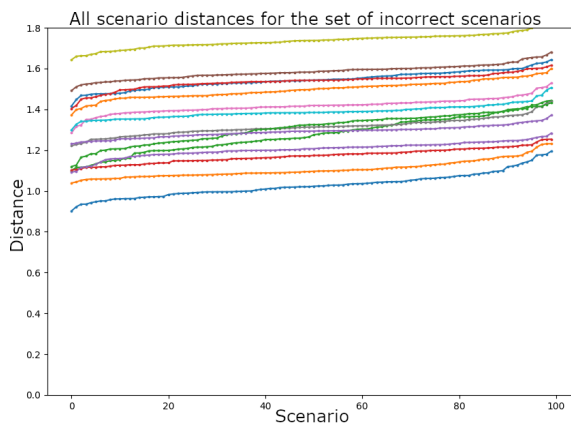


Figure 5.25: Plot illustrating all scenario distances for the set of incorrect scenarios.

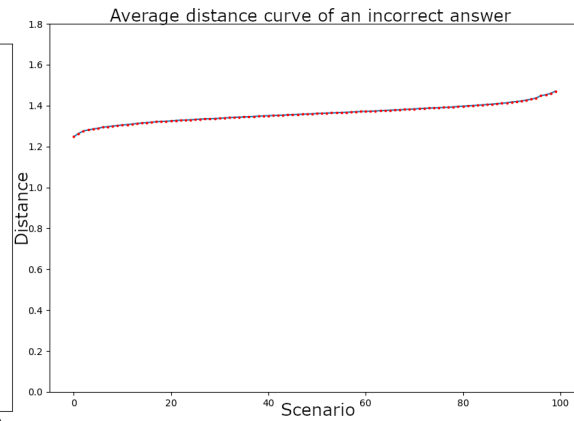


Figure 5.26: Plot representing the distances between a query and the average distances of all scenarios in the set.

Average Incorrect Scenario Metrics	
Average Largest Gap	0.0246
Average Min Distance	1.248
Average Max Distance	1.470
Average Range	0.2226
Average Standard Deviation	0.04446
Average Relative Largest Gap	11.1%

Table 5.9: Table showing the average measured results for the different distance metrics measured from the set of correct scenario answers.

By taking the mean of the two calculated ARLGs, a baseline metric can be calculated.

$$\text{ARLG Baseline} = \frac{26.6\% + 11.1\%}{2} = 18.8\%$$

This metric can now be used as a baseline to measure the solutions performance against.

To evaluate the solution, a set of test queries are needed. The method used to procure these test queries was to extract 10 scenarios at random. These scenarios were then analyzed using an external visualization tool provided by the collaborating company. Through this analysis, queries were constructed for each scenario that should reasonably find these scenarios.

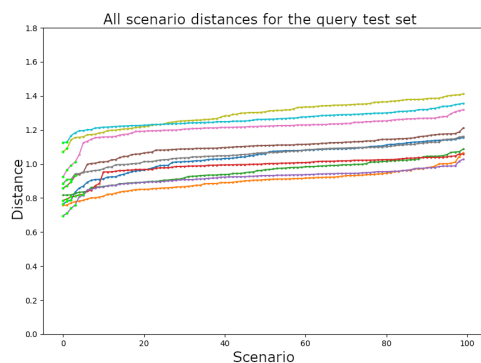


Figure 5.27: Plot illustrating all scenario distances for the query test set.

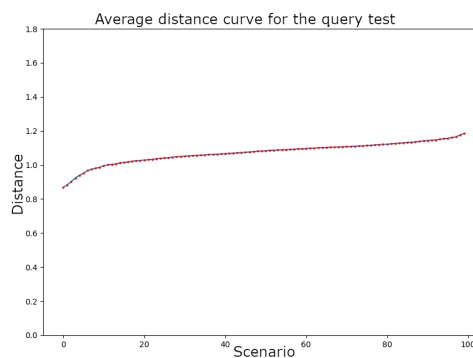


Figure 5.28: Plot representing the distances between a query and the average distances of all scenarios in the query test set.

Average Query Test set Metrics	
Average Largest Gap	0.046
Average Min Distance	0.8685
Average Max Distance	1.185
Average Range	0.3168
Average Standard Deviation	0.06401
Average Relative Largest Gap	14.5%
Answers found by Z-score validation	8/10

Table 5.10: Table showing the average measured results for the different distance metrics measured from the query test set.

By comparing the ARLG measured for the test set, 14.5% against the baseline ARLG 18.8%, it is observable to see that the average characteristics of the test set answers are closer to the characteristics of a query without answers rather than one with correct answers. As the test queries should all contain answers, this is not a desirable outcome. However, this is not due to a poor search functionality of the solution as by manually checking we can see that the closest answers given are indeed either the scenario previously analyzed or a similar scenario that fits the query. However, the issue lies in when queries are too broad, when a query fits for a large share of the scenarios in the dataset, the ARLG will be low and the curve looks similar for a query with no correct answers even though a large number of answers are in fact correct.

If attention is instead diverted to the Z-score validation, this metric reports that 8 of the 10 test queries have correct answers. While this is a much better result and it accurately confirms large majority of the set, it falls prey to the same issue shown by the ARLG, which is where the query is too general and contains too many answers which offsets the whole curve and decreases the chances for outliers.

Qualitative evaluation

The search function of Genius was qualitatively evaluated through manual evaluation. A varied set of queries were tested using Genius and the answers were manually checked and cross-checked with the external visualization tool provided. During this evaluation some key observations were found: (1) If a query described an answer that exists in the set of scenarios, that answer was always one of the first answers in the list returned. (2) The best answer was not always the first returned answer. Sometimes when querying for a specific scenario, other close but not quite the same scenarios get lower distances to the query than the sought after scenario. (3) Querying using words that could have many different meaning based on context can lead to confusion for the search function, e.g. querying for "change lane" will find scenarios that include other instances of "change" and "lane" such as: "The car is driving in the middle of the **lane**. A speed limit **change** is approaching."

The Z-score answer validation evaluation was performed simultaneously as the search function evaluation. After the Z-score validation all answers were manually evaluated to check whether the validation had any false positives or negatives. From this we could observe that very rarely would false positives occur and if an answer was validated it would most likely be apt an answer for the query, conversely, however, it was not uncommon for false negatives to occur, e.g. the Z-score validation would not always find all answers. Where the Z-score validation shines is when the user queries for a very specific scenario, in this case all answers tend to be extreme outliers to the rest of the scenarios.

The few-shot prompting improved the format of the descriptions, each description now follows the same format. However, some outliers still exist, although these outlier descriptions follow the same format these also include an extra section, usually curtailing the rest of the description. These extra segments tend to be superfluous text, summarizes or miscellaneous extrapolation, e.g. the LLM might add a text that there is an increased risk of collision if the rest of the description mention a high traffic load.

The keyword search made it easier to see what scenarios could be correct answers but since it is based on similar words it can sometimes mistake words and show a keyword that should not be found e.g. that it conflates the word "Speeding" with the word "Driving".

6

Conclusion

This chapter discusses the findings from the thesis and summarizes them to answer the research questions; finally, it also includes a discussion of ideas and suggestions for future work.

6.1 Discussion of the results

This discussion section is divided into separate sections for each of the three research questions, where the findings of the cycles are discussed. It also includes a more general discussion of the current implementation of the artifact in terms of capabilities and limitations.

6.1.1 RQ1

We explored how to effectively use the available signal data in vehicle logs. Our implementation depends on the manual selection of a subset of signals that then are converted to text that contains some important information using tools that are also chosen manually per signal basis. This selection of signals and tools per signal is then applied to all vehicle logs that are processed. The reason for this is that all the signals, ~ 6000 , cannot fit inside the limited context size of the LLM that we are using. The Gemma 7b model has a context size of 4096 tokens. The text that is created for each signal is around 10-20 words long, which results in about 7.5-15 tokens, assuming that the number of words is slightly more than the number of tokens. This would require a context size of around 60 000 tokens. This is far beyond what our model currently supports.

Our aim was to select signals that can create a general summary of what happens in the scenario. The selected signals were selected in collaboration with our supervisor at Zenseact. This is, however, a difficult selection to make since the number of scenarios that can arise in a vehicle is obviously very many and might be specifically tied to a signal that was not included in our selection. We selected 16 signals that correspond to basic information such as vehicle speed, what country the vehicle is in, and the current speed limit. Using just 16 signals does put us well below the context limit, but we wanted to make room for few-shot prompting, and the model might not perform at its best while using many prompts, especially when it is quantized.

The set of scenarios that we generated in the evaluation of cycle 1 were specifically

chosen to contain events that are captured in these signals, therefore the results might not have been as clear with a different selection of scenario categories. However, we also utilize information from video footage of the scenario, therefore information that was present in a signal, for example if the car was in a tunnel or not, could have been captured anyway. This fact might have impacted the results of the evaluation form that we sent out to developers at the collaborating company in our favor, since we are using scenarios where the utilized signals are relevant. However, the results of the evaluation show that the descriptions are of a quality that is acceptable and it includes the most important aspects of the scenarios, which shows that the chosen approach has potential.

6.1.2 RQ2

The multimodal LLM we are using to extract information from the videos accept input images one at a time, it is possible to feed images in series in a prompt and to tell the model that they are part of a video. However, due to the processing time of the images, we opted to use just one frame per scenario to keep the processing time as short as possible. Of course, it would increase the amount of information that could be extracted from a scenario if we were to use multiple frames. This was also something that was noted in the comments on the evaluation form that was part of the evaluation in cycle 1, information from later in the scenarios was missing.

The current limited approach succeeds to an admirable degree in capturing the most important aspects of the images. The generated descriptions tend to include weather conditions, road conditions, traffic conditions as well as what type of environment the car is situated in, e.g. urban or country roads. Based on the evaluation form in cycle 1, the information from the images was mostly correct, but often missed that there were other cars in the scenario and sometimes mistook urban environments for rural ones. To answer the question, the most vital environmental factors can be extracted using this approach with a multimodal LLM but smaller details can be missed. The cause of the missing information could be several, it could be due to the heavy quantization, or it could be due to the limited resolution inherent to the model we are using. The model looks at images in a 336x336 pixel resolution.

The multimodal LLM mostly captures information relative to the environment surrounding the vehicle, where the signals capture more complicated information about what the car is experiencing during the scenario. The combination of these two approaches creates a much more detailed and complete description of the scenario as they capture different aspects of it.

6.1.3 RQ3

In cycle 1, an evaluation of vectors of the embedded scenario descriptions were made through the use of t-SNE visualization. This flattens the high dimensional vectors to a vector with just 2 dimensions, this makes it a lot easier to visualize in a plot. However, this operation does not guarantee that relations within the high dimensional data is transferred in a representative way, since we are moving from a

vector of 1024 dimensions to a vector of 2 dimensions. However, even if it is not a perfect representation, it does indicate that the embeddings are effective in the sense that semantically similar scenarios end up close in the vector space. The evaluation is based on a small set of 8 scenarios, with this database the searching works very well. As seen in Figure 5.8 the scenarios are in well defined clusters and a query for a snowy scenario lands in the right place.

Another evaluation of the embeddings is how the vectors relate to each other, to evaluate this we plot a histogram containing the distances between each of the embeddings. In cycle 1, this evaluation does not give that much information since we are only using 8 scenarios, the distribution looks somewhat uniform, but no real conclusion can be drawn from this small dataset. When scaling up the dataset from 8 to 100 scenarios in cycle 2, we instead see a Gaussian distribution emerge. Our initial hope was to see a uniform distribution since that would mean that all the embeddings are evenly spread out in vector space. A Gaussian distribution does however make sense, the scenarios are different from each other, but they are not that different from each other on the scale of language. The scenarios are all closely related in the sense that they all are about driving scenarios and many of the scenarios describe very similar scenarios while some describe more rare scenarios resulting in the Gaussian distribution.

One limitation that is inherent to text embedding models, such as the BGE-large model we are using in this thesis, is that they do not have a good semantic representation of numbers. Therefore, information such as the actual speed of the vehicle will be ineffective to search for semantically.

6.1.4 Artifact discussion

Current implementation of the artifact might not replace searching for scenarios with SQL, but it might instead be a complimentary tool to search for specifics in scenarios that are not simple to find through signal data alone, like for example "snowy scenario in a rural area" or "intersection in an urban area with traffic lights" since that information is just available from the video.

The search functionality of the artifact has been observed to be stable and yields satisfactory results. If during a query, the signals and scenario queried for exist then in our observations the scenario or scenarios similar to the query will be returned first, however not necessarily in the order of most fitting.

Searching for scenarios work well at this larger scale, the results from the previous cycle did transfer to using more scenarios. Harder to find homogeneous scenarios like "lane change" since that scenario is part of many scenarios, also other scenario has similar wording, like "car in opposite lane".

While it was hard to determine without manually checking if answers are correct, the two answer validation methods make it easier. The keyword search method worked remarkably well to catch incorrect answers, as if none of the keywords had semantically similar words in the scenario then, most likely, no scenarios are correct answers to the scenario. It could however struggle to give a clear distinction between

correct and incorrect answers as some key words, such as "lane" or "speed" triggered in every scenario which rendered the keyword search method moot.

The Z-score validation method boasted remarkable results and stability when querying for specific and unusual scenarios. For example if the user queries for an event that happens in a small percentage of the scenarios in the set then the distances to these scenarios will be much smaller than the distances to the rest of the set and the Z-score validation method will find them as outliers. However if the user queries for an event that happens in a large share of the scenarios then the distances to these scenarios will be more common and therefore not necessarily found as outliers, thus the more specific and rarer the scenario queried for is then the likelier it is that the Z-score validation method will find and label the correct answers.

While neither of the two answer validation methods were sure-fire methods that always find and label the correct answers, they are helpful to use in combination with each other to get a quick grasp of the returned scenarios after a query.

Few-shot prompting successfully standardized the descriptions, normalizing the embedding values to ignore writing style. While the LLM maintains the intended format, it sometimes adds an unnecessary section at the end. Another issue is the occasional inclusion of signals mentioned in the few-shot prompts that should not appear in the descriptions. This can lead to irrelevant matches during searches, as descriptions that state a signal wasn't active still match due to the signal's mention. One solution could be to replace actual signal names in the prompts with placeholders like "signal X" or "signal Y" to prevent unintended mentions, though these placeholders might then appear in descriptions.

Increasing the number of prompt examples could improve our method, but we are constrained by model size, quantization, and token limits. Currently, we use two examples to avoid exceeding the token limit. Additionally, the model's small size and quantization might lead to a loss of reasoning in longer prompts, increasing the risk of hallucination.

Our current method entails leveraging a multimodal model to generate a description from the camera footage, and then using an LLM to combine this description with a list of relevant processed signals. This was done to break the tasks into parts and allow an LLM trained purely on text to write the final description. However technically we could have used a multimodal model to generate the final description directly, inputting both image and signal data in one prompt. This would decrease time and calls drastically, however as it stands now with smaller, quantized models we believe that it would have a negative effect on the descriptions. With larger models, and more VRAM at hand, it would be interesting to explore this avenue of using only model and call per description.

6.2 Summary

RQ1: *To what extent can an LLM generate a description based on signals in the form of tabular data?*

Current LLM models can not fully utilize the vast tabular data that was used in this thesis, at the very least not with the smaller size of models that we deployed, instead, some sacrifice will have to be made in what signals the model should utilize. Based on our findings from evaluating the descriptions generated when using a subset of signals, the model can generate descriptions that are acceptable and capture some of the most important aspects of a vehicle scenario.

RQ2: *To what extent can an multimodal LLM with vision describe the scenario a vehicle is in based on images?*

Using a multimodal LLM to describe images is, according to our findings, an effective approach to capture information about the environment that the vehicle is in, in any given scenario.

A multimodal model can describe general information in an image of a vehicle scenario well. However, some detail is often lost, such as other vehicles in the image. This is likely largely due to the limited resolution. In our solution a max resolution of 336x336 pixels is available which means that the images resolution becomes heavily downgraded before being processed by the model.

RQ3: *To what degree can a combination of descriptions from tabular and image data be used to capture the semantics of a vehicle scenario in a vector representation?*

Based on our findings from the smaller scale solution in cycle 1, using an embedding model to embed the semantic meaning of a vehicle scenario works well to group similar scenarios close together.

Some limitations exist in our solution, the text embeddings model good semantic representation of numbers makes querying for number specific scenarios, such as specific speeds very difficult. Another limitation is that similar wording that mean different things will embed close to each other in vector space, making it hard to find scenarios that are not very specific, for example "change lane" will be close to "driving in the left **lane** with an upcoming **change** in speed limit".

The solution shows the best result when the query is specific enough that only a few of the scenarios in the data set can fulfill the query.

6.3 Future Work

To improve the artifacts generated descriptions, one interesting avenue to explore would be to increase the amount of frames used from the camera footage. At the moment, due to model and VM restrictions we only use one frame from the video per description to extract information from. With more VM resources and larger models, one could use more and more frames from the video, ideally every frame

would be used, although this would most likely be unnecessary and quite costly for the performance. Reaching one frame per second however would most likely not be taxing for a heavier model and with more resources at hand and would deliver a lot more interesting information to be used in the description.

At the moment we use the Euclidean distance from the query to the scenarios as the primary searching method, with keywords, Z-score and the ARLG as complementary validators. Potentially these 4 methods could be combined to make up a fitness score, which would serve as the primary method of selecting scenarios relevant to the query. This could be beneficial as sometimes the distance alone is not sufficient as some scenarios which may carry a semantically close meaning to the query may not actually mention the exact same situation which the keyword search method could potentially find, in this instance a scenario with a higher distance but all keywords found would be promoted over scenarios with a lower distance but with fewer or no keywords found. Potentially other methods could be added to this fitness score, where each method added would be weighted with how much effect it has on the final fitness score.

In this thesis we have used models trained for general purpose. A way to improve the search method would be to use a fine-tuned, or even fine-tune our own, embedding model on automotive domain texts. This could allow the model to better see nuances in different scenarios more clearly. As it is now, all scenarios are fairly close to each other in the embedded vector space as all scenarios are relevant to cars and traffic, a model trained specifically for the automotive domain however would not group scenarios together just because of their relevance to the automotive domains as that would be the norm, instead it would find weight in what specifically occurs during the scenario. If fine-tuning is not a viable strategy due to resource restrictions, a way to potentially emulate the same result could be to process the generated descriptions before embedding them to remove words that are general to the automotive domain and not specific to scenarios. For example, words such as "car", "driving", "road" could be removed to bring emphasis to the remaining words that are more specific to the scenario. This could however bring undesired consequences if the general words are important in a given context.

Another way to improve the generated descriptions and allow for more detailed scenarios to be queried for, would be to increase the amount of signals processed and given to the LLM to use in the description. This would require a model with a much larger context size, for example Gemini 1.5, which boasts a token limit of 1 000 000 tokens compared to the 4000 token limit currently used in this thesis. This would allow much more specific signals to be included in the description.

Another idea is to use the current approach with just generated description from images as target (labeled data) to try and train an LLM that can generate a description based on the entire signal table as input. The table would probably need to be encoded in a specific way for that to work. This approach would require a lot more resources and time and it might not deliver a great product as the signal data is not available in the video, so some information such as indicators and alerts would be lost.

Bibliography

- [1] S. Miroslaw, *Automotive Software Architectures : An Introduction*. Springer, 2021, ISBN: 9783030659387. [Online]. Available: <https://search.ebscohost.com/login.aspx?direct=true&db=edsebk&AN=2802601&site=eds-live&scope=site&authtype=guest&custid=s3911979&groupid=main&profile=eds>.
- [2] R. Sharma, “Big data for autonomous vehicles,” in *Deep Learning and Big Data for Intelligent Transportation: Enabling Technologies and Future Trends*, K. R. Ahmed and A. E. Hassanien, Eds. Cham: Springer International Publishing, 2021, pp. 21–47, ISBN: 978-3-030-65661-4. DOI: 10.1007/978-3-030-65661-4_2. [Online]. Available: https://doi.org/10.1007/978-3-030-65661-4_2.
- [3] R. Hussain and S. Zeadally, “Autonomous cars: Research results, issues, and future challenges,” *IEEE Communications Surveys Tutorials*, vol. 21, no. 2, pp. 1275–1313, 2019. DOI: 10.1109/COMST.2018.2869360.
- [4] L. Ziegler, F. Krämer, and N. Haustein, “Ai and big data management for autonomous vehicles,” *ATZelectronics worldwide*, vol. 15, no. 4, pp. 40–45, Apr. 2020, ISSN: 2524-8804. DOI: 10.1007/s38314-020-0180-8. [Online]. Available: <https://doi.org/10.1007/s38314-020-0180-8>.
- [5] T. Risch, “Query language,” in *Encyclopedia of Database Systems*, L. LIU and M. T. ÖZSU, Eds. Boston, MA: Springer US, 2009, pp. 2260–2261, ISBN: 978-0-387-39940-9. DOI: 10.1007/978-0-387-39940-9_1090. [Online]. Available: https://doi.org/10.1007/978-0-387-39940-9_1090.
- [6] R. Lebet, D. Grangier, and M. Auli, *Neural text generation from structured data with application to the biography domain*, 2016. arXiv: 1603.07771 [cs.CL].
- [7] Y. Zhao, H. Zhang, S. Si, L. Nan, X. Tang, and A. Cohan, *Investigating table-to-text generation capabilities of llms in real-world information seeking scenarios*, 2023. arXiv: 2305.14987 [cs.CL].
- [8] P. Li, Y. He, D. Yashar, *et al.*, *Table-gpt: Table-tuned gpt for diverse table tasks*, 2023. arXiv: 2310.09263 [cs.CL].
- [9] X. Chen, B. Mersch, L. Nunes, *et al.*, “Automatic labeling to generate training data for online lidar-based moving object segmentation,” *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6107–6114, Jul. 2022, ISSN: 2377-3774. DOI: 10.1109/lra.2022.3166544. [Online]. Available: <http://dx.doi.org/10.1109/LRA.2022.3166544>.
- [10] T. Amaratunga, *Understanding Large Language Models, Learning Their Underlying Concepts and Technologies*. Springer, 2023, ISBN: 9798868800177. [On-

- line]. Available: <https://link.springer.com/book/10.1007/979-8-8688-0017-7>.
- [11] R. Sennrich, B. Haddow, and A. Birch, *Neural machine translation of rare words with subword units*, 2016. arXiv: 1508.07909 [cs.CL].
- [12] D. Jurafsky and J. H. Martin. “Speech and language processing, 3rd ed. draft.” (2023), [Online]. Available: <https://web.stanford.edu/~jurafsky/slp3/>. (accessed: 15.03.2024).
- [13] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, *Language models are unsupervised multitask learners*, 2019. [Online]. Available: <https://www.semanticscholar.org/paper/Language-Models-are-Unsupervised-Multitask-Learners-Radford-Wu/9405cc0d6169988371b2755e573cc28650d14dfe> (visited on 01/06/2023).
- [14] F. Almeida and G. Xexéo, *Word embeddings: A survey*, 2023. arXiv: 1901.09069 [cs.CL].
- [15] J. Beel, B. Gipp, S. Langer, and C. Breitinger, “Research-paper recommender systems : A literature survey,” *International Journal on Digital Libraries*, vol. 17, no. 4, pp. 305–338, 2016, ISSN: 1432-5012. DOI: 10.1007/s00799-015-0156-0.
- [16] OpenAI, : J. Achiam, *et al.*, *Gpt-4 technical report*, 2023. arXiv: 2303.08774 [cs.CL].
- [17] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, 2023. arXiv: 1706.03762 [cs.CL].
- [18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019. arXiv: 1810.04805 [cs.CL].
- [19] T. B. Brown, B. Mann, N. Ryder, *et al.*, *Language models are few-shot learners*, 2020. arXiv: 2005.14165 [cs.CL].
- [20] P. P. Ray, “Chatgpt: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope,” *Internet of Things and Cyber-Physical Systems*, vol. 3, pp. 121–154, 2023, ISSN: 2667-3452. DOI: <https://doi.org/10.1016/j.iotcps.2023.04.003>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S266734522300024X>.
- [21] H. Liu, C. Li, Q. Wu, and Y. J. Lee, *Visual instruction tuning*, 2023. arXiv: 2304.08485 [cs.CV].
- [22] L. Wang, W. Xu, Y. Lan, *et al.*, *Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models*, 2023. arXiv: 2305.04091 [cs.CL].
- [23] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, *Gptq: Accurate post-training quantization for generative pre-trained transformers*, 2023. arXiv: 2210.17323 [cs.LG].
- [24] Z. Liu, B. Oguz, C. Zhao, *et al.*, *Llm-gat: Data-free quantization aware training for large language models*, 2023. arXiv: 2305.17888 [cs.CL].
- [25] M. Alibeigi, W. Ljungbergh, A. Tonderski, *et al.*, *Zenseact open dataset: A large-scale and diverse multimodal dataset for autonomous driving*, 2023. arXiv: 2305.02008 [cs.CV].

-
- [26] L. van der Maaten and G. Hinton, “Visualizing data using t-sne,” *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008. [Online]. Available: <http://jmlr.org/papers/v9/vandermaaten08a.html>.
- [27] T. Mikolov, K. Chen, G. Corrado, and J. Dean, *Efficient estimation of word representations in vector space*, 2013. arXiv: 1301.3781 [cs.CL].
- [28] A. R. Hevner, S. T. March, J. Park, and S. Ram, “Design science in information systems research,” *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004, ISSN: 02767783. [Online]. Available: <http://www.jstor.org/stable/25148625> (visited on 03/07/2024).
- [29] R. Wieringa, “Design science as nested problem solving,” in *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology*, ser. DESRIST '09, Philadelphia, Pennsylvania: Association for Computing Machinery, 2009, ISBN: 9781605584089. DOI: 10.1145/1555619.1555630. [Online]. Available: <https://doi.org/10.1145/1555619.1555630>.
- [30] R. Wieringa, *Design Science Methodology for Information Systems and Software Engineering*. Jan. 2014, pp. 1–332, ISBN: 978-3-662-43838-1. DOI: 10.1007/978-3-662-43839-8.
- [31] V. Vaishnavi and W. Kuechler, *Design science research in information systems*, 2021. [Online]. Available: <http://www.desrist.org/design-research-in-information-systems/>.
- [32] G. Team, T. Mesnard, C. Hardin, *et al.*, *Gemma: Open models based on gemini research and technology*, 2024. arXiv: 2403.08295 [cs.CL].
- [33] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers, “Mteb: Massive text embedding benchmark,” *arXiv preprint arXiv:2210.07316*, 2022. DOI: 10.48550/ARXIV.2210.07316. [Online]. Available: <https://arxiv.org/abs/2210.07316>.
- [34] S. Xiao, Z. Liu, P. Zhang, and N. Muennighoff, *C-pack: Packaged resources to advance general chinese embedding*, 2023. arXiv: 2309.07597 [cs.CL].
- [35] E. Beeching, C. Fourrier, N. Habib, *et al.*, *Open llm leaderboard*, https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard, 2023.

A

Appendix

A.1 Full descriptions of scenario for model evaluation

Tested scenario was one where an upcoming speed limit change was the prominent event.

A.1.1 Mistral 7b

The car is being tested on a deserted Swedish highway during cloudy and dark conditions. The velocity signal shows a decreasing trend initially, followed by a rising trend, with an average velocity of 15.5 meters/second. An upcoming speed limit change is detected, but no speeding is observed in the given scenario. The car suddenly brakes for a short duration due to a momentary distraction, causing the driver's eyes to leave the road for 2.025 seconds. Road work is detected in one of the lanes, but no immediate threat assessment intervention was noticed. Free space assessment intervention was activated briefly to ensure safety. Overall, the car maintains a moderate speed while navigating the highway during the test drive.

A.1.2 Starling 7b

The car is being driven in a test drive on a straight highway in Sweden with a cloudy, dark sky. The driver briefly looked away from the road, and the car experienced a momentary free space threat assessment intervention. The car experienced an upcoming speed limit change with a mean velocity of 15.539392 meters per second. The brake pedal was pressed for a short duration, and the car was speeding during the test drive, while road work was detected in the lane.

A.1.3 Gemma 7b

The car is driving on a straight highway in Sweden. It is moving at a moderate speed of 15 meters per second and appears to be slowing down. The road ahead has a speed limit change sign and the brake pedal was recently pressed for a brief period. The driver has their eyes on the road but has momentarily looked away for a total of 2 seconds. Road work detected in the lane and there is an upcoming speed limit

change. There is no other notable events or behaviors exhibited by the car during the test drive.

A.1.4 Gemma 2b

The car is driving down a highway in a cloudy and dark atmosphere. The car appears to be moving at a moderate speed, and there are no notable events or behaviors exhibited by the car during the test drive.

A.2 Prompts

A.2.1 Standard prompts

A.2.1.1 Multi-modal LLM Prompt (LLaVA 1.5 7b)

```
"role": "user",
```

```
"content": 'Given an image taken from the dash camera of car during a test drive, generate a concise and detailed description of the scenario the car is in. Your output should provide a clear and comprehensive overview of the driving conditions, environmental factors, and any notable events or behaviors exhibited by the car during the test drive. Use only information from gathered from the image. This information will be used by engineers to understand the scenario that the car is in.'
```

A.2.1.2 LLM Prompt (Gemma 7b)

```
"role": "user",
```

```
"content": 'Given a list of signals and a description of what the cars camera shows during a test drive with a car, generate a concise description of the scenario the car is in, mentioning all signals given as well as all using the information from the image description. Your output should provide a clear overview of the driving conditions, environmental factors, and any notable events or behaviors exhibited by the car during the test drive. Try to give a qualitative description of the scenario along with the quantitative description. Remember to keep it concise and use only the following information delimited by - - - - or #####: The signals are delimited by - - - and the image description is delimited by #####.'
```

A.2.2 Few-shot prompt (Gemma 7b)

```
"role": "user",
```

```
"content" : 'Given a list of signals and a description of what the cars camera shows during a test drive with a car, generate a concise description of the scenario the car is in, mentioning all signals given as well as all using the information from the image description. Your output should provide a clear overview of the driving conditions, environmental factors, and any notable events or behaviors exhibited by the car during the test drive. Try to give a qualitative description of the
```

scenario along with the quantitative description. Remember to keep it concise and use only the following information delimited by - - - - or #####: The signals are delimited by - - - - and the image description is delimited by #####. - - - -Signal descriptions: ### name: Country Unit: unitless Description: Car is in United States of America ##### name: speed limit Unit: meters per second Description: Speed limit of 10 meters per second. name: velocity Unit: meters per second Description: This signal has a Rising trend. The signal has a mean velocity of 14.972377 meters per second. ##### name: speed limit change Unit: unitless Description: There is an upcoming speed limit change. ##### name: drivers eyes on road Unit: unitless Description: The driver looked away from the road for 0.600000 seconds. ##### name: map sensor mismatch detected Unit: unitless Description: A map sensor mismatch was detected. ##### name: free space threat assessment intervention Unit: unitless Description: The free space threat assessment intervention was activated for 0.075000 seconds. ##### name: speeding Unit: unitless Description: The car was speeding. ###- - - -##### The image shows a car driving down a street with a clear blue sky above. The car is positioned in the middle of the street, and there are no other vehicles visible in the image. The car is driving under a traffic light, which is located on the right side of the street. The street appears to be empty, and there are no pedestrians or obstacles in the scene. The car seems to be driving smoothly and following the traffic rules. #####'

"role": "assistant",

"content" : ""**Driving conditions:** The car is driving down a clear street with a clear blue sky above. The road is smooth and there is no visible traffic or pedestrians. The car is positioned in the middle of the lane and appears to be driving smoothly and following the traffic rules.

Environmental factors:

- **Country:** United States of America - **Speed limit:** 10 meters per second - **Velocity:** Ca 15 meters per second

Notable events or behaviors:

- **Speeding:** The car is speeding. - **Map sensor mismatch:** A map sensor mismatch was detected. - **Free space threat assessment intervention:** The free space threat assessment intervention was activated for 0.075000 seconds. - **Drivers eyes off road:** The driver looked away from the road for 0.600000 seconds. - **Speed limit change:** There is an upcoming speed limit change. ""

"role": "user",

"content": "Given a list of signals and a description of what the cars camera shows during a test drive with a car, generate a concise description of the scenario the car is in, mentioning all signals given as well as all using the information from the image description. Your output should provide a clear overview of the driving conditions, environmental factors, and any notable events or behaviors exhibited by the car during the test drive. Try to give a qualitative description of the

scenario along with the quantitative description. Remember to keep it concise and use only the following information delimited by - - - - or #####: The signals are delimited by - - - - and the image description is delimited by #####. - - - -Signal descriptions: ### name: Country Unit: unitless Description: Car is in Sweden ##### name: velocity Unit: meters per second Description: This signal has a Decreasing followed by a rising trend trend. The signal has a mean velocity of 17.820877 meters per second. ##### name: off ramp Unit: unitless Description: There is an off ramp. ##### name: brake pedal pressed Unit: unitless Description: The brake pedal was pressed for 5.125000 seconds. ##### name: drivers eyes on road Unit: unitless Description: The driver looked away from the road for 3.725000 seconds. ##### name: pde warning enabled Unit: unitless Description: A pde warning was triggered. ##### name: road work detected Unit: unitless Description: Road work detected in lane. ##### name: map sensor mismatch detected Unit: unitless Description: A map sensor mismatch was detected. ##### name: free space threat assessment intervention Unit: unitless Description: The free space threat assessment intervention was activated for 16.300000 seconds. ##### name: speeding Unit: unitless Description: The car was speeding. ##### name: lane change Unit: unitless Description: The car changed lane. ###- - -##### The image shows a car driving down a highway during a test drive. The car is surrounded by trees, indicating that it is in a rural or suburban area. The car is moving at a moderate speed, and there are no other vehicles visible in the image. The sky appears to be cloudy, suggesting that the weather might be overcast or potentially rainy. The car's dash camera captures this scene, providing a comprehensive view of the driving conditions and the car's performance during the test drive. #####"

"role": "assistant",

"content" : ""**Driving Conditions:**

The car is driving on a highway in Sweden. The weather is overcast or potentially rainy. The road surface is smooth and dry. Traffic is light, with only the car and no other vehicles visible in the image. The car is surrounded by trees, indicating that it is in a rural or suburban area.

Environmental Factors:

- **Country:** Sweden - **Velocity:** Mean speed of 17.8 meters per second

Notable Events or Behaviors:

- **Speeding:** The car is speeding. - **Brake pedal pressed:** The brake pedal was pressed for 5.125 seconds. - **Drivers eyes on road:** The driver looked away from the road for 3.725 seconds. - **Free space threat assessment intervention:** The free space threat assessment intervention was activated for 16.3 seconds. - **Lane change:** The car changed lane. - **Off ramp:** There is an off ramp present - **Pde warning enabled:** Yes - **Road work detected:** Yes - **Map sensor mismatch detected:** Yes""

"role": "user",

"content": Given a list of signals and a description of what the cars camera shows during a test drive with a car, generate a concise description of the scenario the car is in, mentioning all signals given as well as all using the information from the image description. Your output should provide a clear overview of the driving conditions, environmental factors, and any notable events or behaviors exhibited by the car during the test drive. Try to give a qualitative description of the scenario along with the quantitative description. Remember to keep it concise and use only the following information delimited by - - - - or #####: The signals are delimited by - - - - and the image description is delimited by #####. Follow the format: ****Driving Conditions: **** ****Environmental Factors: **** ****Notable Events or Behaviors: **** Add nothing else."

A.3 Evaluation form

Evaluation of semantic descriptions of scenarios in vehicle logs (5)

We are currently working on our master's thesis here at Zenseact on exploring a more intuitive way of searching for scenarios in the vehicle logs. We would like your help to evaluate the current state to be able to improve the implementation.

We will present you with generated descriptions of two vehicle scenarios along with the corresponding ZVS links of 30 seconds as a reference.

Description of ratings:

1 star: Poor description, several important aspects missing.

2 star:

3 star: Acceptable description of the scenario, capturing some of the important aspects.

4 star:

5 star: Great description, all important aspects of the scenario is captured in the description.

Questions 2-3 (and 6-7) are only applicable if you did not rate the description 5 stars.

* Required

* This form will record your name, please fill your name.

Scenario 1

1. How well does the description capture what scenario the vehicle is in?

Description:

ZVS Link:

*



2. Are there any factual errors in the description of the scenario?

3. Is there anything important missing from the description? For instance, any specific signals missing that you think would improve the description.

4. Other comments.

Figure A.1: Evaluation form page 1

Scenario 2

5. How well does the description capture what scenario the vehicle is in?

Description:

ZVS Link: *



6. Are there any factual errors in the description of the scenario?

7. Is there anything important missing from the description? For instance, any specific signals missing that you think would improve the description.

8. Other comments.

Figure A.2: Evaluation form page 2

A.4 Queries and results

A.4.1 Queries with correct answers:

Query

"Snowy highway in Sweden"

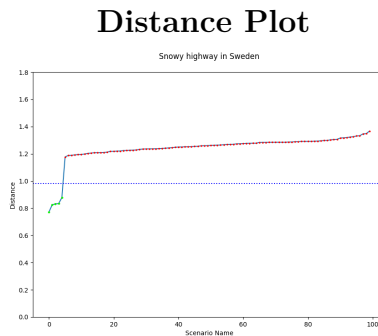


Table of Metrics

Largest Gap	0.2996
Min Distance	0.7713
Max Distance	1.3668
Range	0.5956
Standard Deviation	0.1032
Relative Largest Gap	50.31
Z-Score Answers	5

Query

"Freeway in Sweden"

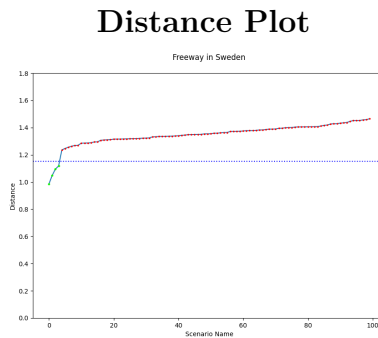


Table of Metrics

Largest Gap	0.1177
Min Distance	0.9844
Max Distance	1.4649
Range	0.4804
Standard Deviation	0.0791
Relative Largest Gap	24.49
Z-Score Answers	4

Query

"Sweden"

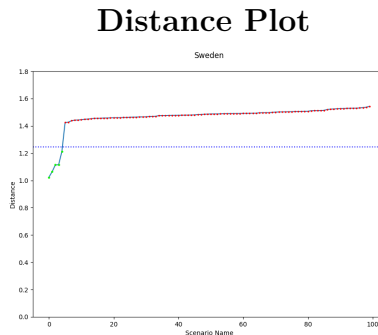


Table of Metrics

Largest Gap	0.2134
Min Distance	1.0235
Max Distance	1.5430
Range	0.5196
Standard Deviation	0.0885
Relative Largest Gap	41.08
Z-Score Answers	5

Query

"Snow on road"

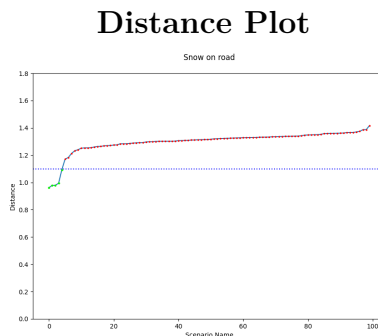


Table of Metrics

Largest Gap	0.0990
Min Distance	0.9615
Max Distance	1.4169
Range	0.4553
Standard Deviation	0.0806
Relative Largest Gap	21.74
Z-Score Answers	5

Query
"Driving under
bridge"

Distance Plot

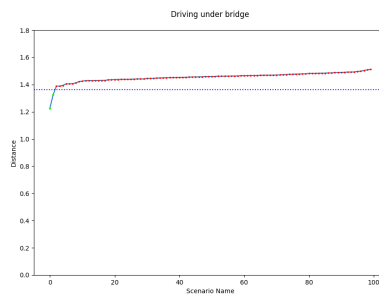


Table of Metrics

Largest Gap	0.0996
Min Distance	1.2261
Max Distance	1.5119
Range	0.2858
Standard Deviation	0.0369
Relative Largest Gap	34.86
Z-Score Answers	2

Query
"Bridge in sce-
nario"

Distance Plot

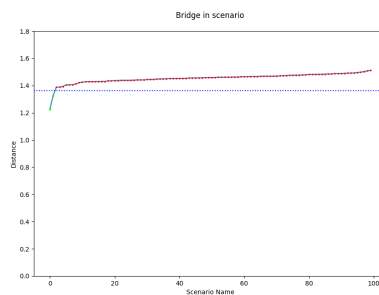


Table of Metrics

Largest Gap	0.0996
Min Distance	1.2261
Max Distance	1.5119
Range	0.2858
Standard Deviation	0.0369
Relative Largest Gap	34.86
Z-Score Answers	2

Query
"Intersection"

Distance Plot

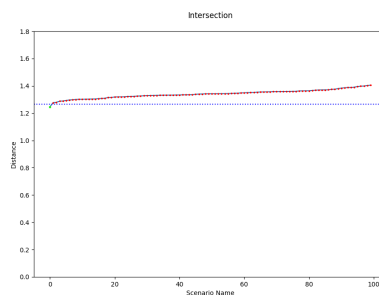


Table of Metrics

Largest Gap	0.0301
Min Distance	1.2467
Max Distance	1.4051
Range	0.1584
Standard Deviation	0.0304
Relative Largest Gap	18.99
Z-Score Answers	1

Query
"Driving through
intersection"

Distance Plot

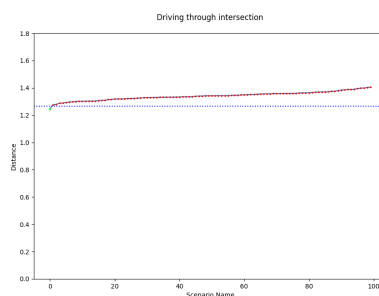


Table of Metrics

Largest Gap	0.0301
Min Distance	1.2467
Max Distance	1.4051
Range	0.1584
Standard Deviation	0.0304
Relative Largest Gap	18.99
Z-Score Answers	1

Query
"Parked Car"

Distance Plot

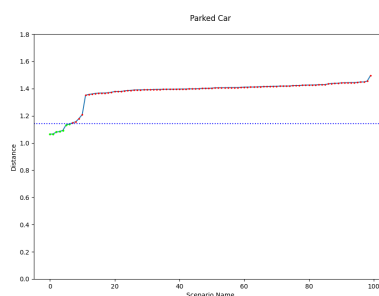


Table of Metrics

Largest Gap	0.1412
Min Distance	1.0646
Max Distance	1.4953
Range	0.4307
Standard Deviation	0.0938
Relative Largest Gap	32.78
Z-Score Answers	7

Query
"United States of America"

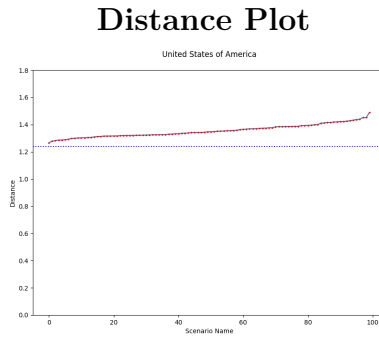


Table of Metrics	
Largest Gap	0.0371
Min Distance	1.2667
Max Distance	1.4891
Range	0.2224
Standard Deviation	0.0459
Relative Largest Gap	16.68
Z-Score Answers	0

Query
"Germany"

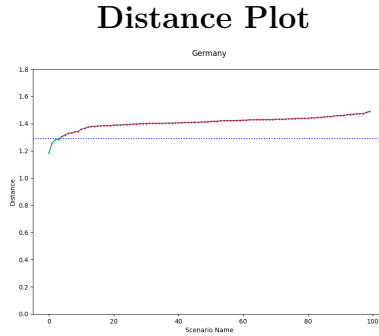


Table of Metrics	
Largest Gap	0.0714
Min Distance	1.1855
Max Distance	1.4886
Range	0.3032
Standard Deviation	0.0478
Relative Largest Gap	23.57
Z-Score Answers	4

Query
"Germany at night"

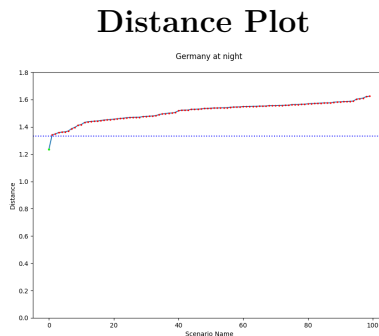


Table of Metrics	
Largest Gap	0.1057
Min Distance	1.2355
Max Distance	1.6250
Range	0.3894
Standard Deviation	0.0717
Relative Largest Gap	27.15
Z-Score Answers	1

Query
"England"

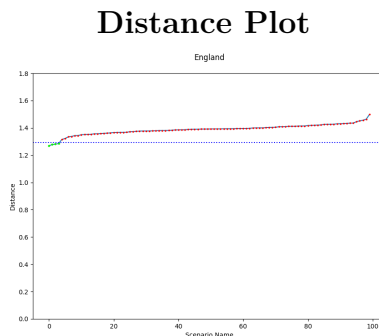


Table of Metrics	
Largest Gap	0.0346
Min Distance	1.2687
Max Distance	1.4984
Range	0.2296
Standard Deviation	0.0378
Relative Largest Gap	15.07
Z-Score Answers	4

Query
"Parked car in the united states"

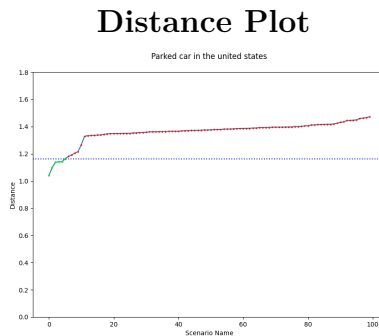


Table of Metrics	
Largest Gap	0.0612
Min Distance	1.0406
Max Distance	1.4720
Range	0.4315
Standard Deviation	0.0792
Relative Largest Gap	14.19
Z-Score Answers	6

Query
"Increased risk of collision"

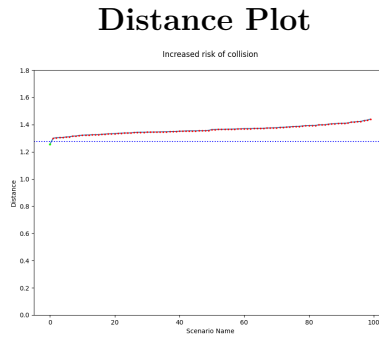


Table of Metrics

Largest Gap	0.0451
Min Distance	1.2549
Max Distance	1.4403
Range	0.1854
Standard Deviation	0.0341
Relative Largest Gap	24.31
Z-Score Answers	1

A.4.2 Queries with no answers:

Query
"Car speeding"

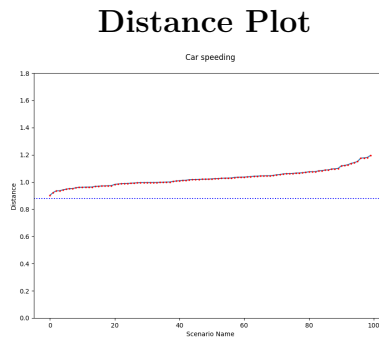


Table of Metrics

Largest Gap	0.0253
Min Distance	0.9018
Max Distance	1.1951
Range	0.2932
Standard Deviation	0.0596
Relative Largest Gap	8.64
Z-Score Answers	0

Query
"Driver looking away"

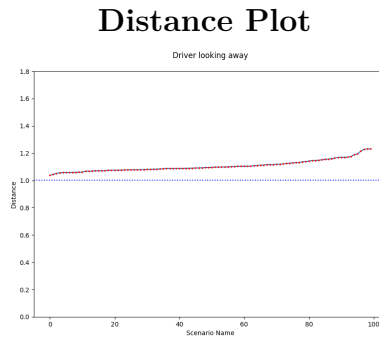


Table of Metrics

Largest Gap	0.0196
Min Distance	1.0388
Max Distance	1.2315
Range	0.1927
Standard Deviation	0.0420
Relative Largest Gap	10.15
Z-Score Answers	0

Query
"Free space threat assessment intervention"

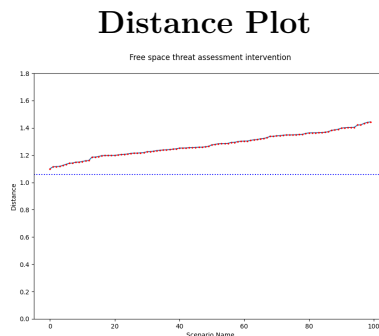


Table of Metrics

Largest Gap	0.0255
Min Distance	1.1000
Max Distance	1.4438
Range	0.3439
Standard Deviation	0.0866
Relative Largest Gap	7.40
Z-Score Answers	0

Query
"Find road edge interventions"

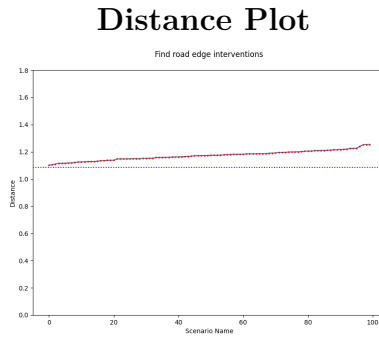


Table of Metrics	
Largest Gap	0.0137
Min Distance	1.1026
Max Distance	1.2538
Range	0.1511
Standard Deviation	0.0350
Relative Largest Gap	9.09
Z-Score Answers	0

Query
"Road work"

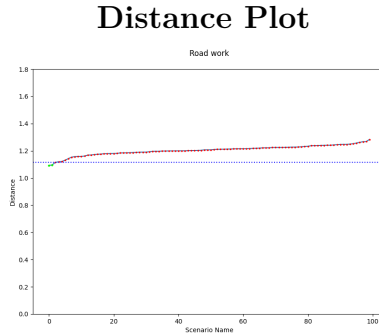


Table of Metrics	
Largest Gap	0.0184
Min Distance	1.0921
Max Distance	1.2835
Range	0.1913
Standard Deviation	0.0361
Relative Largest Gap	9.59
Z-Score Answers	2

Query
"LKA intervention"

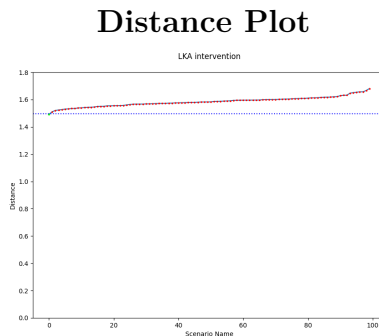


Table of Metrics	
Largest Gap	0.0194
Min Distance	1.4914
Max Distance	1.6810
Range	0.1896
Standard Deviation	0.0351
Relative Largest Gap	10.22
Z-Score Answers	1

Query
"Norway"

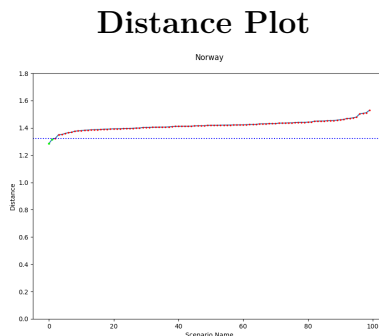


Table of Metrics	
Largest Gap	0.0319
Min Distance	1.2845
Max Distance	1.5291
Range	0.2447
Standard Deviation	0.0378
Relative Largest Gap	13.05
Z-Score Answers	2

Query
"Car crash"

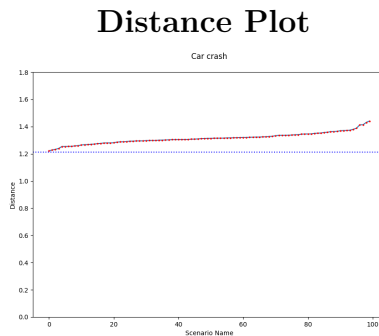


Table of Metrics	
Largest Gap	0.0254
Min Distance	1.2217
Max Distance	1.4408
Range	0.2191
Standard Deviation	0.0415
Relative Largest Gap	11.58
Z-Score Answers	0

Query
"Monaco"

Distance Plot

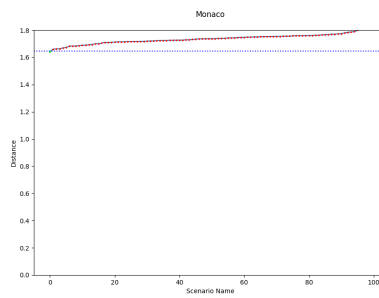


Table of Metrics

Largest Gap	0.0175
Min Distance	1.6426
Max Distance	1.8279
Range	0.1853
Standard Deviation	0.0355
Relative Largest Gap	9.44
Z-Score Answers	1

Query
"Driving across a border"

Distance Plot

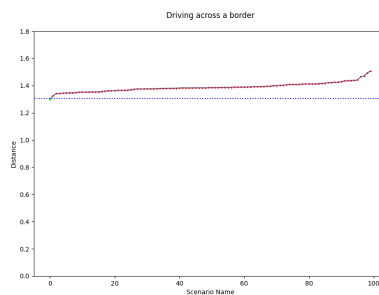


Table of Metrics

Largest Gap	0.0281
Min Distance	1.2990
Max Distance	1.5065
Range	0.2075
Standard Deviation	0.0327
Relative Largest Gap	13.55
Z-Score Answers	1

Query
"Driving through tunnel"

Distance Plot

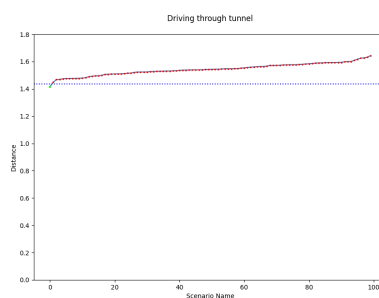


Table of Metrics

Largest Gap	0.0317
Min Distance	1.4160
Max Distance	1.6435
Range	0.2275
Standard Deviation	0.0436
Relative Largest Gap	13.95
Z-Score Answers	1

Query
"Upcoming tunnel"

Distance Plot

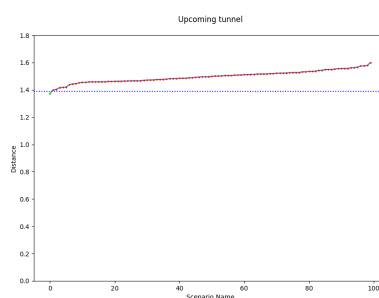


Table of Metrics

Largest Gap	0.0263
Min Distance	1.3723
Max Distance	1.6001
Range	0.2278
Standard Deviation	0.0430
Relative Largest Gap	11.56
Z-Score Answers	1

Query
"Traffic accident"

Distance Plot

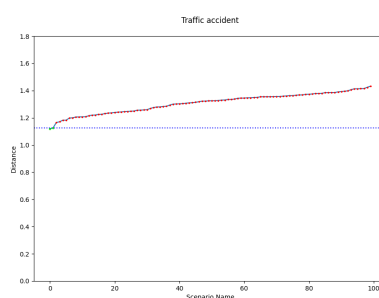


Table of Metrics

Largest Gap	0.0389
Min Distance	1.1197
Max Distance	1.4321
Range	0.3124
Standard Deviation	0.0725
Relative Largest Gap	12.44
Z-Score Answers	2

Query
"Russia"

Distance Plot

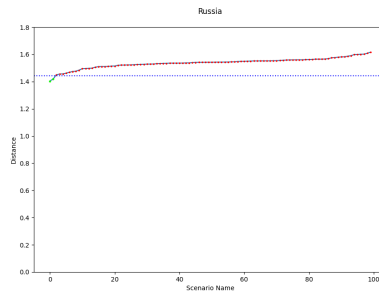


Table of Metrics

Largest Gap	0.0306
Min Distance	1.4040
Max Distance	1.6160
Range	0.2120
Standard Deviation	0.0377
Relative Largest Gap	14.44
Z-Score Answers	2

Query
"Left turn on a country road"

Distance Plot

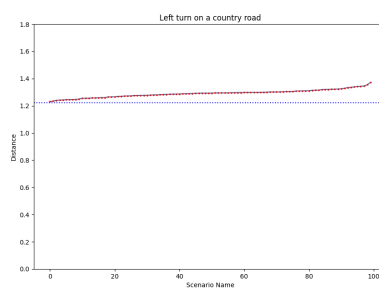


Table of Metrics

Largest Gap	0.0171
Min Distance	1.2308
Max Distance	1.3716
Range	0.1407
Standard Deviation	0.0279
Relative Largest Gap	12.17
Z-Score Answers	0

A.4.3 Test Queries:

Query
"In a parking lot at night, dark and many other parked cars."

Distance Plot

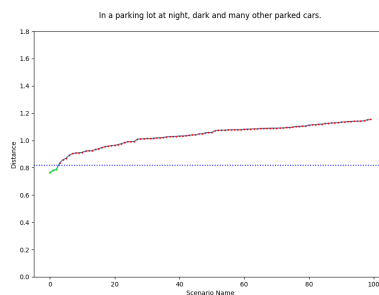


Table of Metrics

Largest Gap	0.0473
Min Distance	0.7644
Max Distance	1.1544
Range	0.3900
Standard Deviation	0.0878
Relative Largest Gap	12.14
Z-Score Answers	3

Query
"Driving on a straight highway at night, not many other cars. Changing lane and overtaking a semi-truck."

Distance Plot

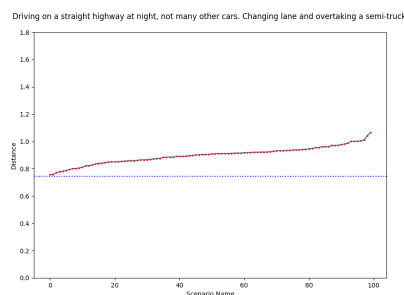


Table of Metrics

Largest Gap	0.0326
Min Distance	0.7570
Max Distance	1.0660
Range	0.3090
Standard Deviation	0.0623
Relative Largest Gap	10.54
Z-Score Answers	0

Query

"Urban environment at night, intersections with green traffic lights. No other cars."

Distance Plot

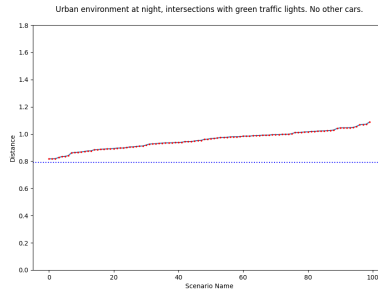


Table of Metrics

Largest Gap	0.0222
Min Distance	0.8174
Max Distance	1.0875
Range	0.2701
Standard Deviation	0.0656
Relative Largest Gap	8.23
Z-Score Answers	0

Query

"Standing still in a parking lot during the day. Some other cars."

Distance Plot

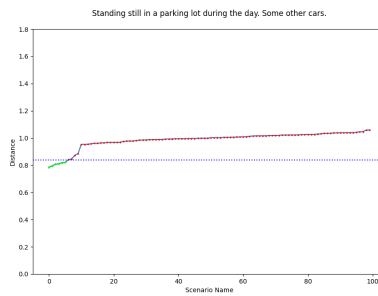


Table of Metrics

Largest Gap	0.0680
Min Distance	0.7852
Max Distance	1.0594
Range	0.2742
Standard Deviation	0.0594
Relative Largest Gap	24.79
Z-Score Answers	6

Query

"Driving on a very snowy country road with trees beside the road, no other cars around. Daytime."

Distance Plot

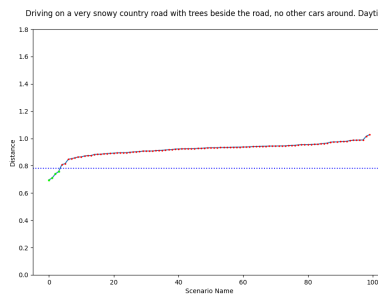


Table of Metrics

Largest Gap	0.0503
Min Distance	0.6960
Max Distance	1.0277
Range	0.3317
Standard Deviation	0.0554
Relative Largest Gap	15.18
Z-Score Answers	4

Query

"Snowy road , dark and cloudy residential area"

Distance Plot

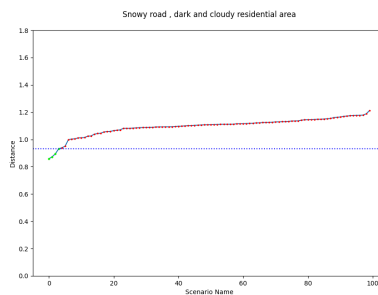


Table of Metrics

Largest Gap	0.0477
Min Distance	0.8584
Max Distance	1.2114
Range	0.3530
Standard Deviation	0.0651
Relative Largest Gap	13.53
Z-Score Answers	4

Query

"Snowy road , country lane , day"

Distance Plot

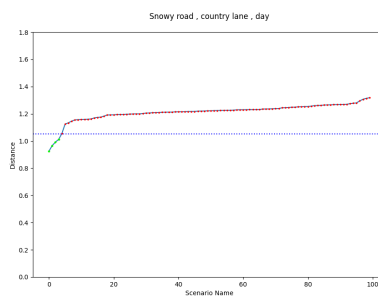


Table of Metrics

Largest Gap	0.0699
Min Distance	0.9240
Max Distance	1.3182
Range	0.3941
Standard Deviation	0.0640
Relative Largest Gap	17.73
Z-Score Answers	4

Query

"Driving at night on a freeway, many cars in front."

Distance Plot

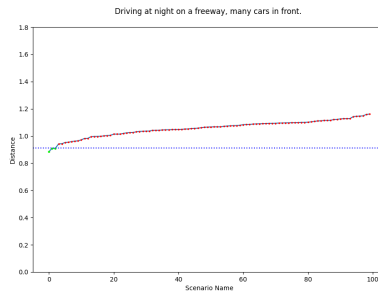


Table of Metrics

Largest Gap	0.0322
Min Distance	0.8848
Max Distance	1.1614
Range	0.2766
Standard Deviation	0.0583
Relative Largest Gap	11.62
Z-Score Answers	3

Query

"Driving at night in the city, empty road"

Distance Plot

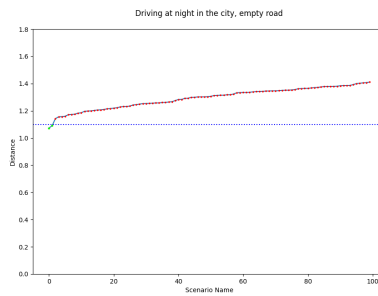


Table of Metrics

Largest Gap	0.0511
Min Distance	1.0726
Max Distance	1.4109
Range	0.3384
Standard Deviation	0.0773
Relative Largest Gap	15.11
Z-Score Answers	2

Query

"Driving on a sunny day in america, car changes lane"

Distance Plot

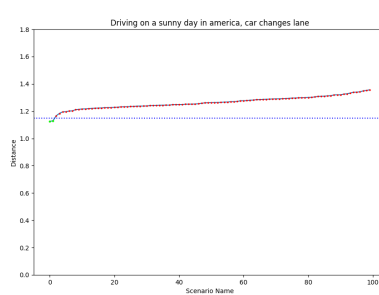


Table of Metrics

Largest Gap	0.0367
Min Distance	1.1252
Max Distance	1.3562
Range	0.2310
Standard Deviation	0.0450
Relative Largest Gap	15.89
Z-Score Answers	2