

# 1 Fine-tuning dataset creation

The fine-tuning process requires a well-structured dataset tailored to the task. To construct this dataset, multiple examples were generated, along with modified distance matrix to ensure variability in training.

## 1.1 Distance Matrix Modification

To prevent the model from memorizing the specific Points of Interest used in the real-world study conducted in Milan, the distance matrix was modified by mapping each POI from the Milan dataset to randomly selected POIs in Rome. The walking durations remained unchanged, as the primary objective was to train the model on the reasoning process rather than the accuracy of the walking durations. Example of Rome matrix:

Capitoline Museums, Museo - Vatican Museums, Museo time: 7 min  
Capitoline Museums, Museo - Via Appia Antica, TargaCommemorativa time: 10 min  
Casa di Goethe, Museo - Barberini, PermatadellaMetropolitana time: 12 min  
Casa di Goethe, Museo - Galleria Borghese, GalleriaArte time: 9 min  
Casa di Goethe, Museo - Monument to Saint Peter, Monumento time: 6 min

In addition to the primary Milan-to-Rome mapping, additional generalized matrices were introduced to further diversify the dataset and enhance model generalization. These matrices preserved the same structural logic but granted a more variable training for the model. The mapping process ensured a realistic and varied distribution of POIs while maintaining consistency in walking durations. Example of generalized matrix:

From metro: GrandCentral, to art\_gallery: MetroGallery - Walking time = 5 min  
From museum: ScienceMuseum, to monument: PoetryGarden - Walking time = 5 min  
From metro: UnionSquare, to theater: Shakespeare - Walking time = 4 min  
From metro: TimesSquare, to theater: Trianon - Walking time = 4 min  
From metro: UnionSquare, to sculpture: WindSpirit - Walking time = 9 min

The result are different matrix consisting of 666 rows, preserving the same POI types but replacing their names. This modification ensures that the model learns the logical structure of itinerary selection rather than memorizing the spatial patterns of Milan. The methodology used for mapping POIs is detailed in the Table 1.

## 1.2 Itinerary Validation and Reasoning

The dataset includes both valid and invalid itineraries, structured as follows:

- A prompt containing all the rules that a valid itinerary must follow (e.g., ending at a transportation hub, avoiding duplicate POIs, retrieving time from the connection list, bidirectionality between POIs, etc.).
- A distance matrix derived from the original correct matrix with intentional modifications.
- An itinerary, which was either correct or incorrect.

The response for each example indicated whether the itinerary was valid or not, along with a detailed explanation of its correctness or errors. This approach ensured that training focused on the reasoning process, helping the model learn to evaluate itineraries based on predefined rules and justify its assessments accordingly. To illustrate the structure of the data set, the following is an example of a prompt and its corresponding response. Example Prompt-response:

PROMPT:  
Validate these itineraries respecting the following rules:

[RULES]  
1. START: Starts with a non-transport location  
2. CONTINUITY: Each stop must connect to the next  
3. TRANSPORT END: Every itinerary MUST end at a transport location  
4. NO INTERMEDIATE TRANSPORT: Transport locations only allowed as final stop  
5. LENGTH: Itineraries can have 3-6 stops

6. UNIQUE: No location can be used twice in the same itinerary  
7. DISTINCT: Intermediate stops should be different across itineraries  
8. BIDIRECTION: The connections are bidirectional  
9. WALKING TIME: Walking time must be retrieved from the connection list

[BIDIRECTIONAL CONNECTIONS - IMPORTANT]  
Every connection in this system works both ways:  
1. If you can walk from A \to B in X minutes  
2. You can ALWAYS walk from B \to A in the same X minutes  
3. Example from the connections list:  
- If "Museum: Louvre \to Church: Notre Dame = 10 min" exists  
- Then "Church: Notre Dame \to Museum: Louvre = 10 min" is automatically valid  
4. Key Points:  
- Same walking time in both directions  
- No need for explicit listing of reverse routes  
- Use ANY connection in reverse with confidence  
- The walking time stays EXACTLY the same in either direction

[CONNECTION LIST]  
From sculpture: WindSpirit, to museum: CityHistory - Walking time = 5 min  
From museum: NaturalHistoryMuseum, to monument: VictoryColumn - Walking time = 4 min  
From monument: VeteransMemorial, to theater: Trianon - Walking time = 5 min  
From museum: ModernArtMuseum, to monument: VictoryColumn - Walking time = 4 min  
.....

Itinerary 6:  
Start: (CityHistory, museum)  
Stop 1: (FreedomTower, monument) - Walking Time: 7 min [reverse]  
Stop 2: (Trianon, theater) - Walking Time: 7 min [forward]  
Stop 3: (TimesSquare, metro) - Walking Time: 4 min [reverse]  
Total Time: 18 min

Response:  
The itinerary is valid. It respects all rules:  
1. ✓ Starts with a non-transport location  
2. ✓ All stops are connected according to the connection list  
3. ✓ Ends at a transport location  
4. ✓ No transport locations used as intermediate stops  
5. ✓ Has 4 stops (within 3-6 range)  
6. ✓ No location is used more than once  
7. ✓ Intermediate stops are different from other itineraries  
8. ✓ Uses bidirectional connections correctly  
9. ✓ Walking times match the connection list

## 1.3 POI-Based Queries

Additional examples focused on retrieving specific POIs, either in general or filtered by POI type (e.g., museums, sculptures, monuments, etc.). The dataset structure for these queries was as follows:

- A prompt explaining key constraints, including the bidirectionality of walking time between POIs (i.e., the walking duration from A to B is the same as from B to A).
- A rule stating that walking times cannot be modified but must be retrieved from the predefined connection list.
- A modified distance matrix incorporating variations for testing retrieval accuracy.
- A query requesting the walking duration between two specific POIs.

The corresponding response provided the correct walking time between the two POIs, ensuring that the model learned to extract accurate duration values based on the predefined constraints.

Example prompt-response:

PROMPT:  
Every connection is automatically valid in both directions:  
If: Museum A - Church B = 10 minutes  
Then: Church B - Museum A = 10 minutes (automatically valid)

Critical Rules for Bidirectional Movement:  
- Same Exact Time: Walking time is identical in both directions  
- Automatic Validity: No need for explicit listing of reverse routes  
- Connection Requirement: Both POIs must appear in the same connection  
- Time Preservation: Times cannot be modified when reversing direction

Available connections:  
Each connection below works in BOTH directions with the SAME time:  
metro: 'Central' - metro: 'Market' time: 12 min  
monument: 'Heritage' - park: 'Memorial' time: 11 min  
metro: 'Union' - park: 'Community' time: 1 min  
park: 'Central' - museum: 'NaturalHistory' time: 10 min  
.....

Question: What's the duration of the walk between park: 'Central' and park: 'Riverside'?

RESPONSE:  
The walking time between park: 'Riverside' and park: 'Central' is 12 minutes.

## 1.4 Connection Retrieval Tasks

The final type of prompt-response pair generated for the training dataset focused on retrieving all connections containing a selected

Milan POI	Mapped Rome POI
Teatro Gerolamo	Teatro dell’Opera di Roma
Museo Teatrale alla Scala	Museo Nazionale Romano
Sala d’armi	Castel Sant’Angelo
Disco in forma di rosa del deserto	Ara Pacis
Monument to Federico Borromeo	Monument to Giuseppe Garibaldi
Monument to Giuseppe Parini	Monument to Cavour
Plaque to Francesco Carmagnola	Plaque to Giuseppe Mazzini
Museo Mangini Bonomi	Capitoline Museums
Duomo	Termini
Cordusio	Repubblica
Missori	Barberini

**Table 1:** Mapping of Milan POIs to Rome POIs for Distance Matrix Modification

POI. The dataset was structured as follows:

- A prompt outlining the critical rules, including bidirectionality of connections, correct time retrieval, and type-based filtering.
- A modified connection matrix incorporating variations for testing retrieval accuracy.
- A query requesting all available walking connections for a given POI

The response contained only the correctly retrieved connection options from the connection list, ensuring that the model accurately extracted all valid links while adhering to the predefined constraints.

Prompt-Response example:

```
PROMPT:
Find connections by EXACT string match.

[CRITICAL RULES]
- List ALL connections where the POI appears EXACTLY as specified
- Include the FULL connection string with both locations and time
- Check BOTH locations in each connection
- Remember connections work in BOTH directions
- When filtering by type, check the type of the OTHER location
- When filtering by time, match the exact duration

[AVAILABLE CONNECTIONS]
Ara Pacis, Scultura - Barberini, FermataDellaMetropolitana time: 11 min
Ara Pacis, Scultura - Borghese Gallery, Museo time: 2 min
Ara Pacis, Scultura - Capitoline Museums, Museo time: 9 min
.....

Question: What are all the walking connections available from 'Doria Pamphilj Gallery, Museo'?

RESPONSE:
Searching for connections containing exactly 'Doria Pamphilj Gallery, Museo'.
Found 36 matching connections:
Teatro dell'Opera di Roma, Teatro - Doria Pamphilj Gallery, Museo time: 7 min
Monument to Vittorio Emanuele II, Monumento - Doria Pamphilj Gallery, Museo time: 1 min
Museo Nazionale Romano, Museo - Doria Pamphilj Gallery, Museo time: 3 min
Castel Sant'Angelo, Scultura - Doria Pamphilj Gallery, Museo time: 0 min
Ara Pacis, Scultura - Doria Pamphilj Gallery, Museo time: 0 min
```

This approach introduced variability in the LLM’s training process, improving its ability to generalize across different scenarios. These datasets were generated using Python scripts, incorporating random and rule-based modifications. The final dataset consisted of 5,500 examples, ensuring a balanced mix of valid and invalid itineraries to refine the model’s reasoning capabilities.

Each entry followed a two-column structure: [prompt, response]. To make the dataset compatible with the LLM, it was converted into a JSON format suitable for fine-tuning. The structure follows a conversation-style dictionary, where each interaction is stored as:

```
[
  {
    "from": "human",
    "value": "prompt",
  },
  {
    "from": "gpt",
    "value": "response"
  }
]
```

The transformation process was implemented using Python, it was converted into a structured JSON format where each row is stored as a dictionary with a "conversations" key. Finally, the dataset was formatted to be uploaded to Hugging Face for cloud-based deployment, ensuring accessibility across different environments. This step

involved converting the dataset into the Hugging Face Dataset<sup>1</sup> format and pushing it to the repository:

#### Listing 1: Upload Dataset to Hugging Face

```
from datasets import Dataset, DatasetDict
dataset = Dataset.from_list(formatted_data)

dataset = DatasetDict({"train": dataset})

dataset.push_to_hub("matteaneda/path_selection_bidirectionality_and_connection_understanding_master")
```

This process ensures that the dataset is readily available for fine-tuning and further experimentation.

## 2 Training Configuration and Optimization Strategies

The final step before training the fine-tuned Mistral-7B model involves setting up the training configuration and optimization parameters using the SFTTrainer class from the trl library [5]. This class provides a streamlined approach for supervised fine-tuning (SFT) of transformer-based models while integrating seamlessly with Hugging Face’s Trainer API<sup>2</sup> [6].

To optimize the training process, we define key hyperparameters using the TrainingArguments class from transformers, ensuring an efficient balance between performance, memory usage, and training speed.

**Training Configuration:** The SFTTrainer is initialized with the pre-trained Mistral-7B model, its tokenizer, and the training dataset. The dataset column used for training is specified as dataset\_text\_field = "text", while dataset\_num\_proc = 2 enables parallel processing [1]. The sequence length is managed via max\_seq\_length to prevent excessive memory consumption.

To optimize efficiency, packing is set to False, ensuring that each sequence is processed independently without packing multiple short sequences together.

The fine-tuning process is guided by several hyperparameters:

**Batching and Optimization:** The per-device batch size is set to 2 due to GPU memory constraints, while gradient\_accumulation\_steps = 4 accumulates gradients before updating weights, effectively simulating a batch size of 8. The optimizer employs weight\_decay = 0.01 to prevent overfitting [2].

**Learning Rate and Scheduler:** A learning rate of 2e-4 is used with a linear scheduler, and a warm-up phase of 5 steps helps stabilize initial training dynamics [4].

**Precision Handling:** The script checks for bfloat16 (Brain Floating Point 16) support via is\_bfloat16\_supported(). If supported, bf16 is

<sup>1</sup> <https://github.com/huggingface/datasets>

<sup>2</sup> [https://huggingface.co/docs/transformers/en/main\\_classes/trainer](https://huggingface.co/docs/transformers/en/main_classes/trainer)

enabled for better numerical stability and efficiency; otherwise, fp16 is used to reduce memory usage [3].

**Logging and Output:** Training progress is logged every step (logging\_steps = 1), and results are stored in output\_dir = "outputs". Automatic reporting to external tracking tools is enabled with report\_to = "wandb".

With these configurations, the model is now fully prepared for training. The next step is executing the fine-tuning process, leveraging the defined settings to efficiently adapt Mistral-7B to itinerary generation. The LLM model, finally, has been called `Itinerary_Selection_Mistral` and pushed to Hugging Face<sup>3</sup>, allowing unrestricted access and deployment.

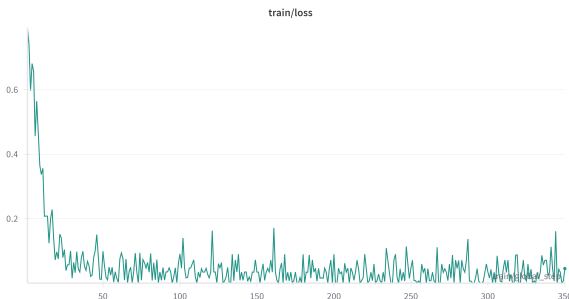
## 2.1 Fine Tune Results

The fine-tuning process was monitored and logged using Weights and Biases (Wandb)<sup>4</sup>, which provided real-time tracking of the training metrics, including loss, gradient norm, and learning rate decay. Wandb’s visualization tools enabled detailed analysis of the model’s performance and facilitated hyperparameter adjustments to enhance convergence.

Metric	Value
Total FLOPs	$2.4286 \times 10^{17}$
Train Epoch	0.50909
Global Steps	350
Gradient Norm	0.04941
Learning Rate	0
Train Loss	0.0453
Final Train Loss	0.05745
Training Runtime (s)	15214.1034
Samples per Second	0.184
Steps per Second	0.023

**Table 2:** Fine-tuning results for the model.

The results indicate a significant reduction in training loss, reaching a final value of 0.0453. This suggests that the model effectively minimized errors throughout the training process. The gradient norm is particularly low at 0.04941, meaning that weight updates were minor, which can be a sign of stable training. Moreover, the learning rate had reached zero by the end of training, which is common when using a scheduler to decay the learning rate gradually. This helps to stabilize the model towards the final steps, preventing sudden fluctuations in performance. The training speed, as indicated by the number of samples processed per second (0.184) and steps per second (0.023), was relatively slow. This could be attributed to the large model size and computational constraints, since the passed dataset for the training was quite heavy for each row (666 connections in each prompt).



**Figure 1:** Training Loss

Lastly, while the overall loss curve demonstrates a downward trend, some fluctuations are observed towards the end. These fluctuations can be explained by several factors. One contributing factor is the decay of the learning rate to zero in the final steps of training. With minimal weight updates occurring, the inherent randomness in mini-batch composition can introduce slight variations in the loss values. Additionally, a relatively small batch size may lead to high variance in gradient updates, making the loss sensitive to individual samples. Some mini-batches may contain more complex examples, leading to minor oscillations in the computed loss. Another explanation lies in the nature of convergence. Once the model has reached an optimal state, large improvements in loss are no longer observed. Instead, the model focuses on refining its learned representations, causing the loss to oscillate around a stable value rather than continuing to decrease. Furthermore, since the model was fine-tuned using Low-Rank Adaptation and 4-bit quantization, numerical precision limitations could have contributed to slight instability in weight updates, amplifying minor variations in loss. These fluctuations do not indicate instability in training but rather reflect the natural variability of optimization dynamics. The model successfully converged with a final low loss value, confirming its ability to minimize errors effectively.

<sup>3</sup> [https://huggingface.co/matteanedda/Itinerary\\_Selection\\_Mistral](https://huggingface.co/matteanedda/Itinerary_Selection_Mistral)

<sup>4</sup> <https://wandb.ai/site/>

tuations can be explained by several factors. One contributing factor is the decay of the learning rate to zero in the final steps of training. With minimal weight updates occurring, the inherent randomness in mini-batch composition can introduce slight variations in the loss values. Additionally, a relatively small batch size may lead to high variance in gradient updates, making the loss sensitive to individual samples. Some mini-batches may contain more complex examples, leading to minor oscillations in the computed loss. Another explanation lies in the nature of convergence. Once the model has reached an optimal state, large improvements in loss are no longer observed. Instead, the model focuses on refining its learned representations, causing the loss to oscillate around a stable value rather than continuing to decrease. Furthermore, since the model was fine-tuned using Low-Rank Adaptation and 4-bit quantization, numerical precision limitations could have contributed to slight instability in weight updates, amplifying minor variations in loss. These fluctuations do not indicate instability in training but rather reflect the natural variability of optimization dynamics. The model successfully converged with a final low loss value, confirming its ability to minimize errors effectively.

## 2.2 Model testing

After fine-tuning, the model was tested to evaluate its ability to generate logical and structured itineraries based on predefined constraints. The testing process involved feeding the model with a structured prompt and analyzing the generated outputs to ensure coherence, logical consistency, and adherence to the given constraints.

### 2.2.1 Model Initialization

The fine-tuned model was loaded using the `AutoModelForCausalLM` class from the Hugging Face Transformers library. The model was quantized in 4-bit format to optimize performance while maintaining accuracy. The corresponding tokenizer was also initialized to ensure proper text tokenization and input formatting.

### 2.2.2 Preprocessing and Data Mapping

The input data consisted of the Points of Interest matrix previously generated, including the walking durations between them. To standardize input, POI types were mapped to a predefined set of categories. Additionally, the matrix was structured starting with 'REGULAR CONNECTION' for connections between cultural POIs such as museums, monuments, etc. while for the metro station connections each row was starting with 'METRO CONNECTION', to increase the understanding of the LLM. Finally, the matrix was shuffled to introduce variability in the generated itineraries.

Original Type	Mapped Type
Museo	museum
FermataDellaMetropolitana	metro_station
GalleriaDArte	art_gallery
Monumento	monument
Scultura	sculpture
Teatro	theater

**Table 3:** Type Mapping for Points of Interest

### 2.2.3 Prompt Design for Testing

A structured prompt was designed to guide the model in generating walking itineraries. The prompt included:

- A clear set of rules ensuring that only valid connections were used.
- Constraints on itinerary length, ensuring each itinerary contained exactly  $n$  POIs before reaching a metro station.
- A systematic step-by-step approach requiring the model to validate each connection.
- A checklist for validation, ensuring each leg of the itinerary was logically consistent.
- The complete list of available connections to ensure the model only used valid transitions (distances matrix).
- The desired output structure of the response to ensure easy post-processing of the responses.

The critical structure of the prompt enforced:

1. The first POI in each itinerary must be a predefined starting point. This is necessary since the model should get a current/predefined starting POI where the people are and not generate the itineraries starting from a random location.
2. Intermediate POIs must not include metro stations since these are not interesting POIs to be visited.
3. The final destination must be a metro station since the final goal is to arrive at a metro station at staggered intervals.
4. The model must explicitly copy connection details, including travel time, without generating new data.
5. Each itinerary must have a unique starting connection, to avoid multiple groups of people visiting the same POI at the same time.

#### 2.2.4 Prompt Structure Analysis

The prompt design follows a multi-layered approach to ensure accurate generation:

1. Initial Task Definition: The prompt begins with a clear statement of the task parameters, defining the number of itineraries required and the number of cultural locations to visit.
2. Critical Rules Section: This section establishes non-negotiable constraints that guide the model's generation process. It emphasizes the importance of exact string matching for connections and the prohibition of metro connections in intermediate legs.
3. Step-by-Step Process: A procedural algorithm is provided for the model to follow at each decision point. This algorithmic approach helps the model maintain consistency throughout the generation process.
4. Verification Mechanisms: The prompt incorporates self-verification steps where the model must confirm that each selected connection adheres to the established rules.
5. Output Formatting: A detailed template is provided for the expected response structure, which facilitates both human readability and automated parsing of results.

#### 2.2.5 Prompt Variables Configuration

The prompt dynamically incorporates several key parameters that control the generation process. These variables allow for flexible experimentation with different scenarios and constraints:

- interval: Time difference (in minutes) between consecutive itineraries. To ensure the staggered arrival at the metro stations.
- numb\_itineraries: Number of desired distinct itineraries. Thinking that group of people can follow the same itineraries to leave a certain event to avoid congestion in the same POIs

- starting\_poi: Common initial location for all itineraries. Since all the people should start from the same location event.
- numb\_pois\_to\_be\_visited: Number of POIs to visit during each itinerary including the last stop at a metro station.
- additional\_informations: Optional field for supplementary instructions or constraints.
- allowed\_types: Permitted POI categories (can be empty, 'Teatro', 'Scultura', 'Monumento', 'GalleriaDArte', 'Museo'). To simulate a filtering based on the users preferences.

#### 2.2.6 Model Output and Validation

The model was queried using the designed prompt, responses were generated using a maximum token limit to capture complete itineraries. The generated itineraries were then analyzed based on logical consistency, ensuring each step followed the constraints; correctness, verifying that the connections used matched the provided dataset; and diversity, ensuring multiple unique itineraries were produced. If any generated output violated the rules, adjustments were made to the prompt to refine the model's adherence to the constraints. The results demonstrated that the model was capable of following complex instructions while maintaining coherence in its generated itineraries.

### 3 Gamification Additional Information

#### 3.1 Itinerary results

The model 'Itinerary\_Selection\_Mistral' has been runned to generate the three itineraries; The response generated from the model has been represented in the bullet point below:

##### • Itinerary #1

**Leg 1:** Museo del Novecento → Collections of Ospedale Maggiore 9 min

**Leg 2:** Collections of Ospedale Maggiore → Royal Palace of Milan 7 min

**Leg 3:** Royal Palace of Milan → Monument to Federico Borromeo 8 min

**Leg 4:** Monument to Federico Borromeo → Cordusio metro 4 min

##### • Itinerary #2

**Leg 1:** Museo del Novecento → Museo Poldi Pezzoli 8 min

**Leg 2:** Museo Poldi Pezzoli → Monument to Giuseppe Missori 11 min

**Leg 3:** Monument to Giuseppe Missori → Disco in forma di rosa del deserto 10 min

**Leg 4:** Disco in forma di rosa del deserto → Cordusio metro 6 min

##### • Itinerary #3

**Leg 1:** Museo del Novecento → Leonardo3 Museum 6 min

**Leg 2:** Leonardo3 Museum → Monument to Cesare Beccaria 7 min

**Leg 3:** Monument to Cesare Beccaria → Monument to Saint Ambrose 7 min

**Leg 4:** Monument to Saint Ambrose → San Babila metro 2 min

### 3.2 MCQs results

The generic model 'mistral-7b-instruct-v0.3-bnb-4bit' has been employed to generate the questions about the above listed POIs:

- **Collections of the Ospedale Maggiore**

*When was the Ospedale Maggiore established?*

- A) 1456-01-01T00:00:00Z (\*).
- B) 1457-01-01T00:00:00Z.
- C) 1455-01-01T00:00:00Z.

*Fact: The Ospedale Maggiore was established around the year 1456.*

- **Royal Palace of Milan**

*Which of the following is true about the Royal Palace of Milan?*

- A) It is a cultural centre and home to international art exhibitions. (\*)
- B) It was originally designed to include two courtyards but these were later dismantled to make room for the Duomo.
- C) It was abandoned for over two years after World War II and its condition further deteriorated.

*Fact: More than 1,500 masterpieces are on display annually at the Royal Palace of Milan.*

- **Monument to Federico Borromeo**

*Who was the architect of the Monument to Federico Borromeo?*

- A) Costantino Corti (\*).
- B) Leonardo da Vinci.
- C) Michelangelo.

*Fact: The Monument to Federico Borromeo was built around 1865.*

- **Museo Poldi Pezzoli**

*When was the Museo Poldi Pezzoli established?*

- A) 1881 (\*).
- B) 1846.
- C) 1850.

*Fact: The Museo Poldi Pezzoli is notable for its broad collection of Northern Italian and Netherlandish/Flemish artists.*

- **Monument to Giuseppe Missori**

*Who was the creator of the Monument to Giuseppe Missori?*

- A) Riccardo Ripamonti (\*).
- B) Giuseppe Missori.
- C) Leonardo da Vinci.

*Fact: The Monument to Giuseppe Missori was built around 1916 and is made out of bronze.*

- **Disco in forma di rosa del deserto**

*Who created the Disco in forma di rosa del deserto, an Italian national heritage sculpture?*

- A) Arnaldo Pomodoro (\*).
- B) Leonardo da Vinci.
- C) Michelangelo.

*Fact: The Disco in forma di rosa del deserto was created by Arnaldo Pomodoro around 1990.*

- **Leonardo3 Museum**

*What is the main theme of Leonardo3 Museum in Milan?*

- A) The life and works of Leonardo da Vinci as an artist and inventor (\*).
- B) The history of the Galleria Vittorio Emanuele II.
- C) The Italian Renaissance period.

*Fact: Leonardo3 Museum is located at Piazza della Scala, Milano, Italy.*

- **Monument to Cesare Beccaria**

*Who was the architect of the Monument to Cesare Beccaria?*

- A) Giuseppe Grandi (\*).
- B) Leonardo da Vinci.
- C) Michelangelo.

*Fact: The Monument to Cesare Beccaria is considered Italian national heritage.*

- **Monument to Saint Ambrose**

*Who was the architect of the Monument to Saint Ambrose in Milan?*

- A) Luigi Scorzini (\*).
- B) Pope Paul VI.
- C) Saint Ambrose.

*Fact: The Monument to Saint Ambrose in Milan was created by Luigi Scorzini, an Italian sculptor.*

The above listed questions are generated based on the input provided to the LLM and have been checked for correctness.

### 3.3 Experiment Results

Group 1, preferring museums, theaters, and monuments, followed the generated itinerary:

- Start: Museo del Novecento
- Stop 1: Teatro Gerolamo (5 min)
- Stop 2: Monument to Federico Borromeo (11 min)
- Stop 3: Duomo di Milano Museum (8 min)
- Final Stop: Duomo (Metro station, 5 min)

**Total expected time: 29 min**

The group of three participants executed the itinerary all together and completed it in an average of **31 minutes**, only 2 minutes longer than the estimated time. Participants successfully answered the multiple-choice questions and captured images at each location.

Group 2, selecting only museums and monuments, received the following itinerary:

- Start: Museo del Novecento
- Stop 1: Monument to Cesare Beccaria (6 min)
- Stop 2: Duomo di Milano Museum (5 min)
- Stop 3: Monument to Giuseppe Missori (6 min)
- Final Stop: Missori (Metro station, 1 min)

**Total expected time: 18 min**

The group of four participants completed the itinerary in two distinct groups of 2 people each in an average of **21 minutes**, exceeding the estimated time by 3 minutes.

Group 3 had no restrictions on POI types, resulting in the following itinerary:

- Start: Museo del Novecento

- Stop 1: Monument to Cristina Trivulzio di Belgiojoso (8 min)
- Stop 2: Royal Palace of Milan (8 min)
- Stop 3: Monument to Federico Borromeo (9 min)
- Final Stop: Cordusio (Metro station, 5 min)

**Total expected time: 28 min**

The group of three participants completed the itinerary separately in an average of **33 minutes**, with a deviation of 5 minutes from the expected result. Participants actively engaged with the gamification elements.

The results demonstrate that the estimated travel times were generally accurate, with deviations ranging from 2 to 5 minutes across different groups. The variations can be attributed to factors such as walking speed differences, short pauses for photographs, and brief discussions at POIs.

## References

- [1] J. Devlin, M. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 4171–4186, 2019.
- [2] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *International Conference on Learning Representations (ICLR)*, 2019.
- [3] P. Micikevicius, M. Narang, J. Alben, G. Diamos, E. Eldridge, J. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu. Mixed precision training. *International Conference on Learning Representations (ICLR)*, 2018.
- [4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.
- [5] L. von Werra, Y. Belkada, L. Tunstall, E. Beeching, T. Thrush, N. Lambert, S. Huang, K. Rasul, and Q. Gallou  dec. TRL: Transformer reinforcement learning. <https://github.com/huggingface/trl>, 2020. Accessed: 2025-03-11.
- [6] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush. Transformers: State-of-the-art natural language processing. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, 2020.