TripMind - AI-powered personalized travel experiences

My Anh Tran Nguyen
20235474
Anh.TNM235474

Huyen Nguyen Thu
20235507
Huyen.NT235507

Ha Dan Tran Le
20235483
Dan.TLH235483

Ly Nguyen Khanh
20235600
Ly.NK235600

Ngoc Anh Pham Thi
20235473
Anh.PTN235473

Abstract

This paper presents TripMind, an advanced multi-agent framework for comprehensive travel planning developed through the convergence of supervised learning, reinforcement learning, and generative artificial intelligence. The system is engineered to fundamentally address the limitations regarding data faithfulness and course planning prevalent in existing recommendation tools by leveraging a custom-crawled dataset of authentic community reviews from Google Maps and TripAdvisor. Experimental results confirm that the collaboration between these specialized neural architectures not only enhances recommendation accuracy but also establishes a viable approach to Explainable AI (XAI) in the tourism sector, providing travelers with routes that are both technically optimized and deeply human-centric and trustworthy.

Keywords: Multi-agent systems, Transformer, BiLSTM, Deep Q-Learning, Large language models, Sentiment analysis, Route optimization.

1 Introduction

1.1 Context and motivation

As time goes on, the tourism industry is increasingly adapting to the preferences of Gen Z and Millennials, as they are gradually becoming its main customer base. Modern travelers are moving away from rigid, pre-packaged tours toward "experience-based" and highly personalized journeys. This trend highlights a preference for solo travel and the exploration of hidden gems, authentic local spots that offer a deeper connection to the culture.

However, this evolution has brought about the challenge of information overload. Travelers often spend hours, if not days, on a tedious manual process: searching for destinations, reading hundreds of conflicting reviews to verify quality, and attempting to stitch these locations into a feasible route. While traditional travel agencies and media outlets exist, their content is often criticized for being impoverished, relying on static, censored, or promotional materials that fail to capture the dynamic taste of modern travelers.

As a result of these limitations, many travelers have shifted away from traditional sources and toward digital traveling platforms. Unlike traditional media, the feedback on these platforms is contributed directly by the community. These reviews are unfiltered, authentic, and free from the strict editorial censorship often found in mainstream travel journalism. For solo travelers, these "real-world" insights are the primary criteria for planning a trip. Nevertheless, extracting useful information from this massive, unstructured data remains difficult due to several challenges:

- Semantic gap: Current systems struggle to interpret descriptive natural language queries that contain multiple attributes (e.g., “a quiet cafe with a workspace and plenty of natural light”).
- Quantitative bias: Reliance on star ratings can be misleading as they are easily manipulated. Most systems fail to analyze the actual sentiment within the text to distinguish between genuine quality and fake ratings.
- Lack of itinerary planning: Most applications provide a fragmented list of places without solving the route optimization problem based on real-world geographical coordinates.
- Communication barrier: Output is typically presented as dry data or lists, lacking natural interaction and the ability to explain why a specific plan was recommended.

1.2 Project goals and the multi-agent system

This project introduces TripMind, an advanced Multi-Agent System (MAS) that utilizes Deep Learning to automate the entire "end-to-end" travel planning process. TripMind not only suggests destinations, but it also acts as a sophisticated digital companion that understands user psychology through four specialized agents:

- Agent 1 (Semantic retrieval - Transformer): Utilizes a multi-task Transformer Encoder architecture to comprehend user intent and retrieve the most relevant candidates via semantic search in a 256-dimensional vector space.
- Agent 2 (Quality evaluator - BiLSTM): Employs a Bi-directional Long-Short Term Memory (BiLSTM) network to perform deep sentiment analysis on thousands of community reviews (scraped from Google Maps and TripAdvisor), filtering out results with suspicious ratings and ranking them based on authentic quality.
- Agent 3 (Route optimizer - Deep Q-Learning): Applies Deep-Q Network (DQN) to solve the itinerary planning problem, finding the most efficient and geographically logical route between the selected locations.
- Agent 4 (Natural language generator - DeepSeek 3.1V): Leverages a Large Language Model (LLM) via the DeepSeek 3.1V API to synthesize technical results into a natural, engaging, and personalized travel plan for the end-user.

1.3 Contributions

This project contributes to the field of AI-driven tourism in Vietnam by:

- Developing a specialized dataset: Creating a dedicated pipeline for scraping and processing Vietnamese travel data from community-driven platforms, providing a foundation for local tourism research.
- Integrating diverse learning paradigms: Proposing a seamless coordination mechanism between Supervised Learning (Transformer/BiLSTM), Reinforcement Learning (DQN), and Generative AI (LLM).
- Advancing explainable AI (XAI): Utilizing an LLM to bridge the gap between complex neural network outputs and human understanding, explaining the rationale behind specific recommendations and optimized routes to build user trust.

2 Related works

2.1 Transformer and Self-Attention

Before Google published the seminal paper “Attention Is All You Need” (Vaswani et al., 2017) most Natural Language Processing (NLP) tasks, particularly Machine Translation, relied on Recurrent Neural Network (RNN) (Elman, 1990) architectures. The primary weaknesses of this approach were the difficulty in capturing long-range dependencies between words and slow training speeds caused by sequential input processing. Transformers were developed to address these two core issues.

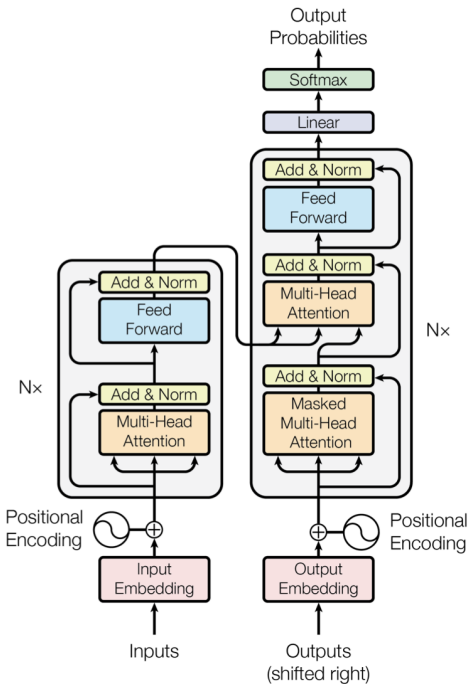

Figure 1: The Transformer architecture

2.1.1 Parallelization

Unlike sequential models such as RNNs or LSTMs, which require data to be processed at each time step t, the Transformer allows for the simultaneous computation of all tokens in a sequence. This eliminates sequential dependency, enabling the model to fully leverage the computational power of modern hardware like GPUs and significantly shorten training time.

2.1.2 Long-range dependencies

In recurrent networks, information must pass through multiple intermediate steps, often leading to the vanishing gradient problem and the loss of semantic information in long sequences. The Transformer fundamentally solves this by establishing direct links between every pair of words in a sentence through the Attention mechanism, regardless of their physical distance. This ensures semantic integrity even in large documents.

The Scaled Dot-Product Attention mechanism allows the model to assign different weights to each word in a sentence based on its semantic correlation with the other words:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

Subsequent research, such as BERT (Devlin et al., 2019), has demonstrated that utilizing Transformer Encoder layers enables the learning of multi-layered language representations. This provides robust support for semantic query tasks and text feature extraction.

2.2 Bidirectional Long Short-Term Memory (BiLSTM)

BiLSTM (Graves and Schmidhuber, 2005) is a variation of LSTMs (Hochreiter and Schmidhuber, 1997) that processes the sequence of inputs in both the forward and reverse directions, which allows the model to have a more complete understanding of temporal dependencies. Figure 2 shows the BiLSTM architecture diagram.

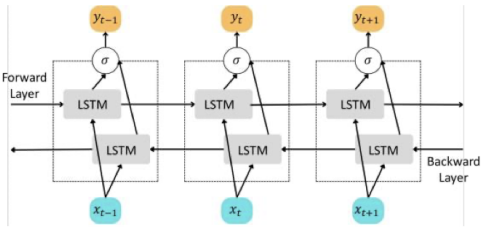

Figure 2: BiLSTM network architecture

Combining information from both the past and the future at each time step enables the model to achieve a deeper understanding of the linguistic context. In Sentiment Analysis tasks, BiLSTM has proven superior in identifying negations or modifiers that significantly impact the overall sentiment of a review.

2.3 Deep Q-Learning (DQN)

Reinforcement Learning (RL) is one of the three primary types of machine learning, alongside Supervised and Unsupervised Learning. At its core, RL relies on a trial-and-error process, where an agent learns from experience through repeated interactions. Major breakthroughs from DeepMind's algorithms, such as AlphaGo and AlphaStar, have demonstrated RL's ability to outperform human intelligence in highly complex strategic games.

2.3.1 Q-tables to neural networks

Instead of using a Q-table, which is limited by massive state spaces, we utilize neural networks to directly approximate the optimal action from a given input state.

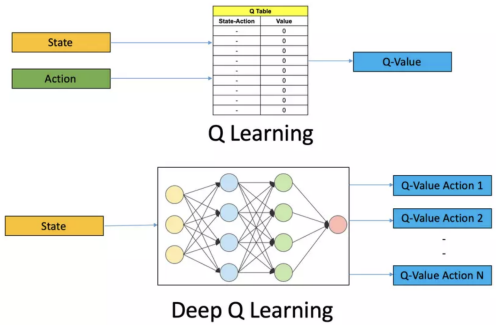

Figure 3: From Q-learning to Deep Q-Learning

2.3.2 Loss function and TD error

To enable the network to "learn", we use a loss function based on Temporal Difference (TD) Error. This measures the discrepancy between the predicted Q-value and the target value derived from the Bellman equation:

$$Loss = \mathbb{E}\left[(\text{Target} - Q(s,a))^2\right]. \quad (2)$$

Optimizing this loss function allows the model to progressively make more accurate decisions with each interaction with the environment.

2.4 Agent AI

The theory of Intelligent Agents (Wooldridge and Jennings, 1995) defines agents as software entities characterized by four core properties: the ability to operate without direct human intervention (Autonomy); the capacity to interact with other agents or humans via a communication language (Social Ability); the ability to perceive their environment and respond timely to changes. (Reactivity); the capacity to take the initiative and exhibit goal-directed behavior (Proactiveness).

In the era of Deep Learning, Agent AI has evolved from simple rule-based systems into sophisticated entities capable of autonomous learning and specialized task execution. The current paradigm is shifting from monolithic models toward Multi-Agent Systems (MAS). In these systems, multiple agents coordinate sequentially or parallelly to solve multi-objective problems, ranging from understanding complex user intent to executing action plans in the real world.

3 Dataset

3.1 Data collection

For our project, we decided to collect reviews on Vietnamese tourist attractions from Google Maps and TripAdvisor. TripAdvisor is one of the largest travel platforms with a large user base of travel enthusiasts. It provides both detailed written opinions and numerical ratings from this user base, which allows for evaluation of not only the overall popularity of a destination, but also its specific pros and cons. Google Maps, on the other hand, has a large and diverse user base, reducing demographic bias and reflecting the consensus of the average person.

Figure 4: TripAdvisor and Google Maps

To collect the data, we used the `playwright` library to dynamically interact with the Google Maps webpage and Omkar's scraping tool to extract data from TripAdvisor, resulting in a dataset of attractions and a dataset of user reviews. Attractions were obtained for each old Vietnamese province, with a maximum of 50 instances per province to avoid scraping noisy data. After processing the attraction dataset, we extracted their corresponding user reviews. To ensure the quality of our input data, we prioritized reviews that are sorted by most detailed.

3.2 Data preprocessing

In this process, to normalize attraction names and retrieve geographic location and coordinates, we used SerpAPI, which provides access to the Google Maps API.

For data cleaning, we removed duplicate records and noisy data from both datasets. In the chance that one tourist attraction is recorded as two different identities, we mapped the attractions' names to their names on Google Maps. We then deleted records sharing the same normalized name, keeping only one of them. For noisy data, we erased any attraction that has 0 reviews and attractions that are not physical destinations, e.g., traveling companies, tours, cruises, and so on. As for the review dataset, we excluded those that contain nonsense and those that are not in Vietnamese. Self-promotional texts were also omitted, since they do not reveal information about the destination. Next, we corrected inconsistent province information by querying the destination names in Google Maps to retrieve their locations and replaced provinces in the dataset with newly established Vietnamese provinces, making our data up-to-date for user queries.

After the data cleaning step, after integrating the data from both sources, we merged the data on attractions and reviews into a unified dataset. As a result, the final review dataset not only includes review-specific information, but also the corresponding metadata on the attraction.

We then perform feature selection to reduce redundancy, only keeping features that are informative for analysis. Both the users' stay dates at the reviewed destinations and the dates when reviews were published are presented in the TripAdvisor data; however, they are typically close in time, so only the published dates were retained. Additionally, we discarded the titles of the reviews, as they are often generic, and their contents are already conveyed through the ratings and the review texts.

Finally, geographic coordinates of destinations were extracted through Google Maps queries to help our model calculate an optimal traveling route.

3.3 Data analysis

The final dataset consists of reviews containing attraction-related information. Each record includes a unique review ID, the review text, the rating ranging from 1 to 5, the review's published date, and the associated trip type (e.g., family, solo, or business). In addition, information on the attraction is also provided, including an ID, the normalized attraction name, its coordinates, the province where it's located, and a list of associated categories. The attraction's province is represented by

both an ID and its name, allowing for province-based filtering and retrieval when users search for travel information. The same logic applies to the categories field: each category includes an ID and a name so that users can explore attractions based on their personal interests.

Our final dataset includes over 14000 reviews of 965 attractions in 34 Vietnamese provinces. The number of attractions per province is as in Figure 5. The disparity comes from differences in tourism development of provinces. The fact that many old provinces were aggregated into one also plays a role in the variance in attraction counts.

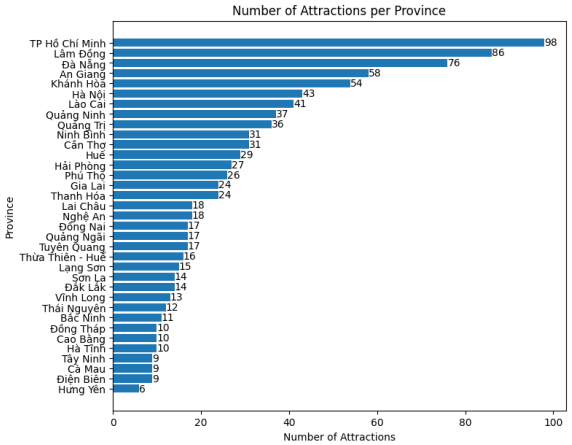

Figure 5: Number of attractions per province

Our dataset includes 122 different categories. The most common one is the general category, tourist attractions & scenic spots, followed by more specific categories, such as religious sites, beaches, historical sites, etc., as shown in Figure 6, which includes the top 20 categories with the most attractions.

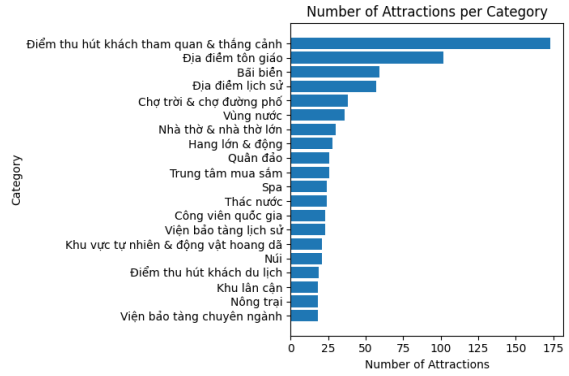

Figure 6: Number of attractions per category

Studying the number of reviews per attraction, we see that the histogram in Figure 7 is skewed: most attractions have relatively few reviews, while a small number of attractions have many reviews. The highest bars are clustered around 10-15 reviews per attraction, suggesting that the majority of attractions receive a modest number of reviews.

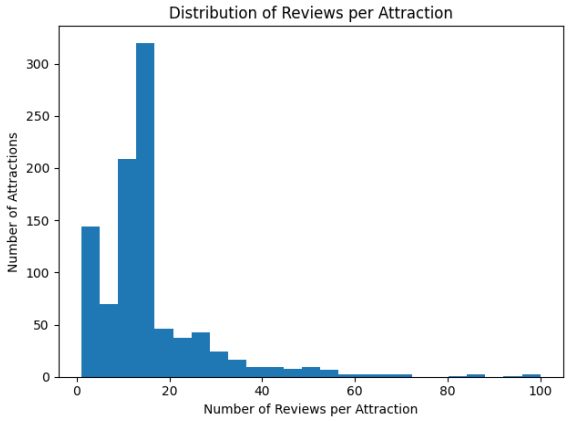

Figure 7: Distribution of reviews per attraction

Considering the recorded trip type, the largest number of reviews fall under trips between friends, followed by "couples" and "families." The least recorded type is business trips, as shown in Figure 8.

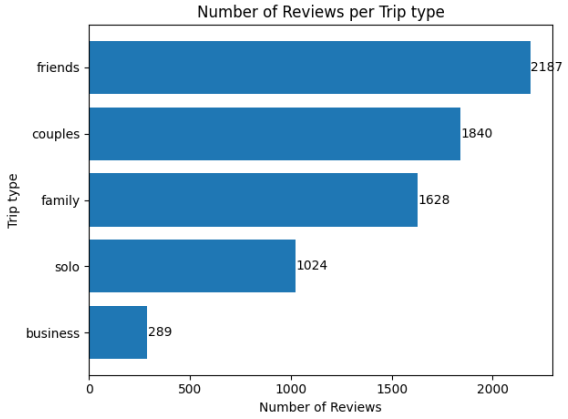

Figure 8: Number of reviews per trip type

Figure 9 shows that the distribution of review text length is skewed: most reviews are short, while very long reviews with over 2000 characters are rare.

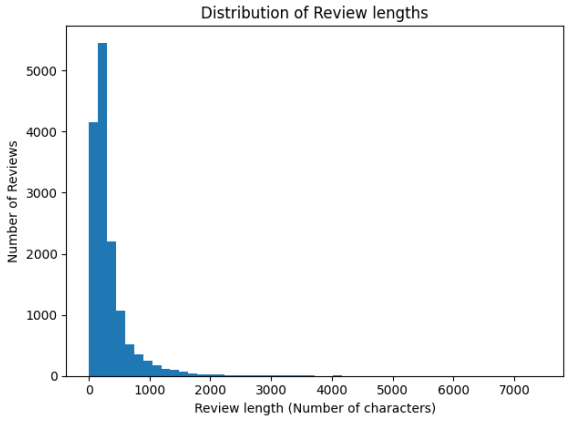

Figure 9: Distribution of review lengths

When we examine the volume of data over time in Figure 10, we recognize a trend that is consistent with real-world behavior: there was a sharp drop during the 2020-2022 timeframe, when people quarantined because of the COVID-19 pandemic. Since people couldn't travel during this time, the number of reviews on tourist attractions decreased drastically, but it picked up and quickly skyrocketed after that.

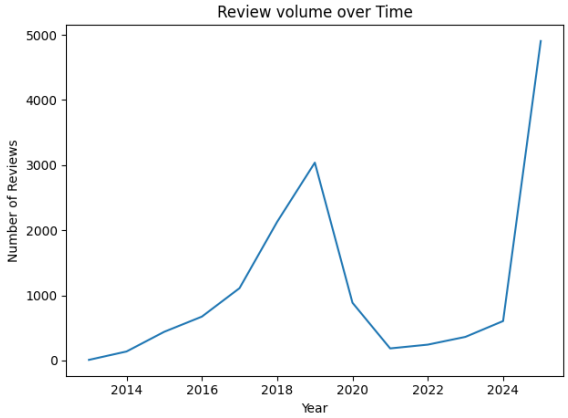

Figure 10: Review volume over time

As we inspect the average numerical ratings (Figure 11) that attractions receive from users and the overall distribution of ratings (Figure 12), we see that people mostly give 5-star ratings. Subsequently, most attractions have average ratings in the higher range as well. This skewed distribution suggests that user ratings may be overly positive, so it is important to investigate other indicators of attraction quality, such as review content, to gain a more nuanced understanding.

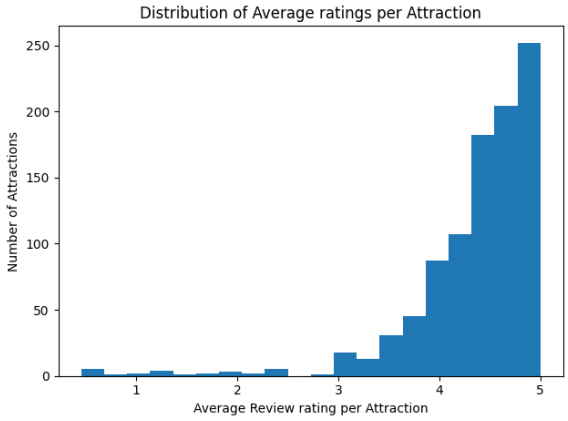

Figure 11: Average rating per attraction

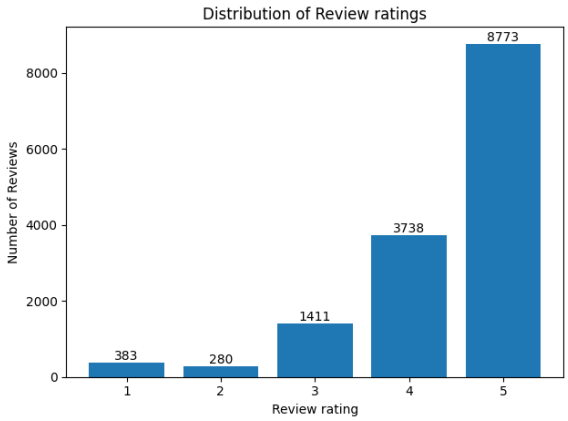

Figure 12: Distribution of ratings

4 Methodology

4.1 Overall architecture

The overall architecture of the TripMind system is designed as an advanced Multi-Agent System that operates through a sequential processing pipeline described in Figure 13 and orchestrated by a central Flask-based API gateway. The workflow begins with Agent 1 (The Matchmaker), which leverages a Transformer Encoder architecture to transform user natural language queries into 256-dimensional feature vectors. By performing a semantic search within a ChromaDB vector database and applying hard filters based on the province by its ID, Agent 1 retrieves a candidate pool consisting of the 15 most relevant attractions. These 15 candidates are subsequently passed to Agent 2 (The Critic), which utilizes a BiLSTM network to conduct deep sentiment analysis on authentic community feedback. This stage meticulously filters the initial pool down to the 5 most suitable locations based on a calculated satisfaction score, effectively mitigating the

impact of biased or low-quality ratings.

The refined list of 5 destinations is then transmitted to Agent 3 (The Strategist), which implements a Deep Q-Network reinforcement learning algorithm to optimize the itinerary. By modeling the travel plan as a sequential decision-making problem, Agent 3 reorders the 5 destinations to minimize the total travel distance and enhance logistical efficiency. Finally, the process concludes with Agent 4 (Natural Language Synthesis), which employs the DeepSeek 3.1V Large Language Model to convert the technical logistical data into a conversational response. This agent generates personalized justifications and travel advice in natural language while strictly adhering to the optimized sequence established by Agent 3, ensuring that the final recommendation is both technically optimized and presented in a human-centric format. The entire coordination facilitates an end-to-end process from initial intent comprehension to natural human-like interaction.

4.2 Agent 1

We designed the architecture of Agent 1 based on the original Transformer structure, specifically fine-tuned for Representation Learning through a multi-task mechanism (Figure 14).

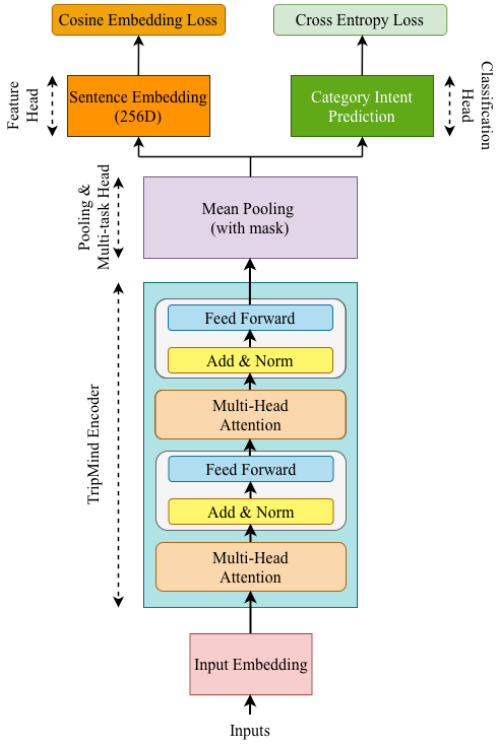

Figure 14: Agent 1's architecture based on the Transformer

4.2.1 Embedding and positional encoding

Word embedding and Scaling: Tokens are converted into continuous vectors using `nn.Embedding` with a dimension of $d_{model} = 256$. Notably, these vectors are multiplied by $\sqrt{d_{model}}$ before adding positional encodings. This step scales the embedding values to ensure that semantic information is not overshadowed by positional data.

4.2.2 Positional encoding

Since Transformers lack inherent recursion, sine and cosine functions are applied to "embed" word order information:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{\text{model}}}}}\right) \tag{3}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{\text{model}}}}}\right) \tag{4}$$

This allows the model to distinguish between sentences with identical words but different sequences.

4.2.3 Transformer encoder layers

Layer structure: The model consists of 6 stacked Encoder layers. Each layer includes two main blocks include of multi-head self-attention (8 heads) and a position-wise feed-forward network (with a hidden dimension of $(d_{model} \times 4 = 1024)$).

Padding masking: The model utilizes `src_key_padding_mask` to instruct the Attention mechanism to ignore `<PAD>` tokens. This prevents padding characters from introducing noise into the attention weights, particularly for short reviews.

4.2.4 Masked mean pooling

Instead of using the traditional `[CLS]` token, the model performs mean pooling across all hidden states in the final layer. This process involves multiplying the Transformer's output by a mask to ensure only actual tokens contribute to the average:

$$\text{Sentence\_Emb} = \frac{\sum(\text{Output} \times \text{Mask})}{\sum \text{Mask}} \tag{5}$$

Applying this technique results in a more generalized and stable representation vector for the entire review or query content.

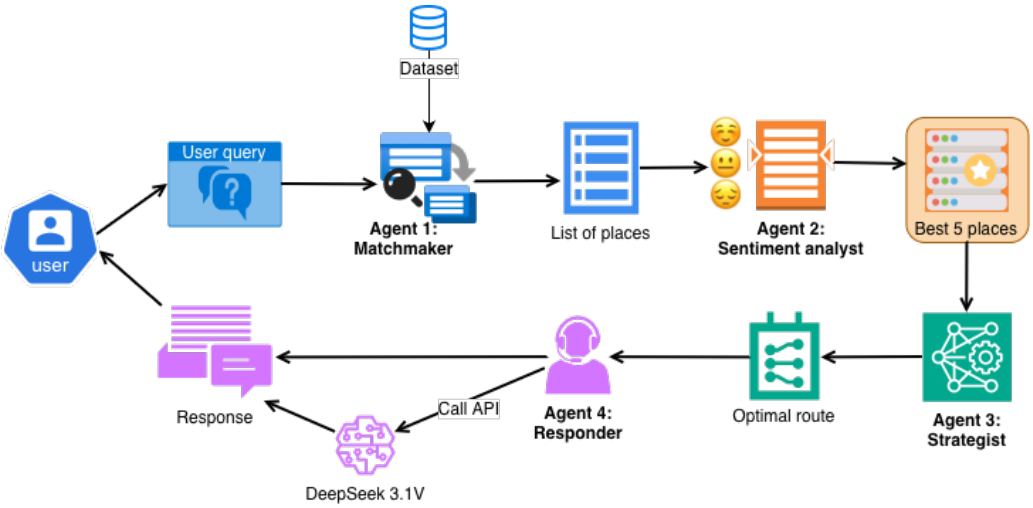

Figure 13: Overall architecture of TripMind

4.2.5 Multi-task loss functions

To enable Agent 1 to perform both accurate semantic search and user intent prediction, the model is trained simultaneously on two objectives.

The first is classification Loss: Cross Entropy Loss (L_{cat}) is used to train the `category_classifier` branch. This forces the model to learn how to categorize locations into the correct "`categories`" field (e.g., "religious sites", "beaches") based on the location "`name`" and user review "`text`".

$$L_{cat} = -\sum_{i} y_i \log(\hat{y}_i) \tag{6}$$

Second is the embedding loss: We utilize Cosine Embedding Loss (L_{emb}) with a $margin = 0.2$. This loss function is pivotal in shaping the vector space:

- If two texts share the same meaning ($y = 1$), the model minimizes the cosine distance between them toward 0.
- If they differ ($y = -1$), the model pushes them apart until the distance is at least equal to the defined margin.

$$L_{\text{emb}}(x_1, x_2, y) = \begin{cases} 1 - \cos(x_1, x_2) & \text{if } y \\ \max\big(0, \cos(x_1, x_2) - \text{margin}\big) & \text{if } y \end{cases} \tag{7}$$

4.2.6 Total objective

The final loss function is a weighted combination of the two components, calculated during each training iteration:

$$Total\_Loss = L_{emb} + 0.5 \times L_{cat} \tag{8}$$

A coefficient of 0.5 is applied to L_{cat} to balance the influence, prioritizing the formation of the vector space for retrieval while still ensuring robust intent classification. This joint optimization allows the model to achieve deeper semantic understanding compared to single-task training.

4.3 Agent 2

After Agent 1 retrieves a list of potential candidates, Agent 2 takes on the role of a "quality assessor." This agent utilizes a deep learning model to perform in-depth sentiment analysis on user reviews, subsequently ranking and filtering for the highest quality and most relevant locations.

4.3.1 Model architecture: BiLSTM sentiment classifier

Agent 2 employs a BiLSTM recurrent neural network to capture the comprehensive context of review sentences. The model is defined within the `SentimentClassifier` class with the following core components in Figure 15:

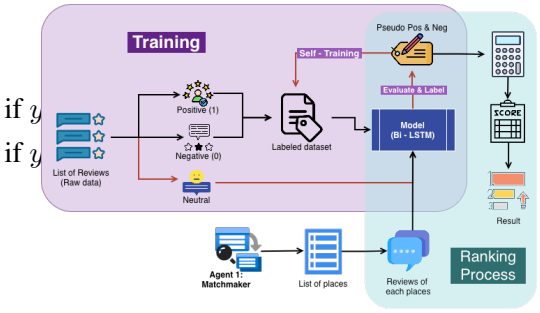

Figure 15: Architecture of Agent 2

Embedding layer: Converts token indices into continuous vector space representations with a fixed dimension $d_{embed} = 128$. This layer helps the model learn semantic relationships between words in the vocabulary.

Bi-directional LSTM Layers: Consists of 2 stacked BiLSTM layers.

- Bi-directionality: Allows the model to process information in both directions: from the beginning of the sentence to the end (forward) and from the end back to the beginning (backward). This is particularly crucial for Vietnamese to correctly interpret negations (e.g., "khong he te" - not bad at all) or intensifiers placed at the end of a sentence.

- Hidden states: Each BiLSTM layer has a hidden size of $d_{hidden} = 256$. At each time step, the output is a concatenation of the forward hidden state (h_t) and the backward hidden state $\overleftarrow{h_t}$, forming a vector $h_t = [h_t; \overleftarrow{h_t}]$ with a size of 512.

Dropout Layer: Applies a Dropout technique with a probability of $p = 0.3$ after the LSTM layers to prevent overfitting during training.

Fully Connected Layer (Output Head): The final hidden state of the second BiLSTM layer (representing the entire sentence) is passed through a Linear layer to project the $512 - dimensional$ space down to the number of sentiment labels (Positive/Negative/Neutral).

4.3.2 Loss Function and Optimization

To train the BiLSTM model for accurate sentiment classification, Agent 2 uses Binary Cross-Entropy with Logits Loss (`nn.BCEWithLogitsLoss`). This choice stems from the binary nature of the task, where the model distinguishes between two primary classes: Positive (label 1) and Non-positive (label 0).

This loss function combines a Sigmoid layer and the Binary Cross-Entropy (BCE) into a single step, providing better numerical stability. The formula for a single sample i is:

$$L_i = -\left[y_i \cdot \log\big(\sigma(x_i)\big) + (1 - y_i) \cdot \log\big(1 - \sigma(x_i)\big)\right] \tag{9}$$

Where:

- x_i: The raw output (logits) from the final Fully Connected layer for sample i.

- y_i: The ground truth label (0 or 1).

- $\sigma(x)$: The Sigmoid activation function, converting logits into the predicted probability $P(y_i = 1)|x_i$

The weights are optimized using the Adam (Adaptive Moment Estimation) algorithm with an initial learning rate of $\eta = 10^{-3}$. Adam was selected for its ability to adapt the learning rate for each parameter, ensuring rapid and efficient convergence.

4.3.3 Ranking process

The workflow for Agent 2 is triggered upon receiving the list of candidate locations from Agent 1 (via the ranking API endpoint):

- Data preparation: Agent 2 receives N locations (defaulting to 15), each accompanied by its specific user reviews. Reviews are cleaned, tokenized, and converted into numerical sequences with a fixed `MAX_LEN = 100`.

- Sentiment prediction: The model is set to evaluation mode (`model.eval()`). Each review passes through the model to calculate the probability $P(positive)$. Each review receives a Sentiment Score in the range [0,1].

- Aggregation and re-ranking: The location score (S_{place}) is calculated as the arithmetic mean of the sentiment scores of all its reviews:

$$S_{place} = \frac{1}{K} \sum_{i=1}^{K} P(positive)_i \tag{10}$$

  (where K is the number of analyzed reviews for that location).

- Final Selection: Locations are sorted in descending order of S_{place}. The Top 5 locations are selected and passed to Agent 3 (Scheduling Agent).

4.3.4 Advantages of the Approach

Using Agent 2 with a BiLSTM model offers significant advantages over relying solely on star ratings: Filtering Fake rating: While many locations have high star ratings due to subjective bias, the text content may contain complaints. BiLSTM "reads

and understands" the actual content for a fairer assessment.

Deep Contextual Understanding: Thanks to the bi-directional mechanism, the model captures subtle linguistic nuances that simple keyword-based methods (like Bag-of-Words) cannot detect.

4.4 Agent 3

Agent 3 is responsible for constructing an optimal visiting order for a fixed-size set of candidate locations provided by Agent 1 and Agent 2. Given a list of five geographically distributed attractions, the objective of Agent 3 is to generate a route that minimizes the total travel distance while ensuring that each location is visited exactly once.

This task is formulated as a sequential decision-making problem, closely related to the Traveling Salesman Problem (TSP), and is addressed using Reinforcement Learning (RL).

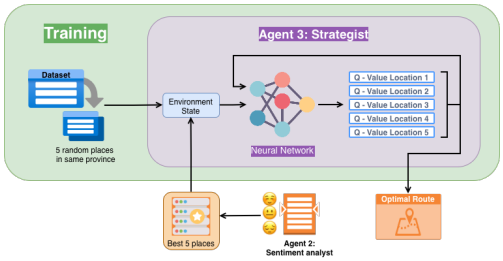

Figure 16: Architecture of Agent 3

4.4.1 Reinforcement learning

Traditional approaches for route optimization, such as brute-force search or heuristic algorithms, can solve small-scale TSP instances but suffer from several limitations:

- They do not generalize across different configurations of locations.
- They are difficult to integrate into a learning-based multi-agent system.
- They cannot adapt dynamically to new constraints or objectives.

Reinforcement Learning offers a flexible framework where an agent learns an optimal policy through interaction with an environment, making it suitable for modeling route planning as a sequence of decisions under constraints.

4.4.2 Q-learning - Initial approach

In the initial phase, the problem was approached using tabular Q-learning, where the action-value function Q(s,a) is explicitly stored in a table.

The Q-learning update rule is defined as:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{11}$$

This method was effective for validating the problem formulation in small, discrete environments. However, it quickly became impractical due to:

- The continuous nature of the state space (latitude and longitude coordinates).
- The exponential growth of state-action combinations.
- Poor generalization to unseen configurations of locations.

As a result, tabular Q-learning was deemed unsuitable for scalable deployment.

4.4.3 Deep Q-learning

To overcome the limitations of tabular methods, the system was extended to Deep Q-Learning (DQN). Instead of storing Q-values explicitly, a neural network is used to approximate the action-value function:

$$Q(s,a;\theta)$$

This transition enables:

- Handling of continuous and high-dimensional state representations.
- Learning generalized routing strategies across multiple episodes.
- Efficient reuse of learned knowledge for unseen location sets.

4.4.4 Objective function

The network is optimized by minimizing the Mean Squared Error (MSE) between predicted Q-values and target Q-values:

$$\mathcal{L} = \mathbb{E}\left[\left(Q(s,a) - \left(r + \gamma \max_{a'} Q'(s',a') \right) \right)^2 \right] \tag{12}$$

where Q' denotes the target network.

4.4.5 Inference and deployment

During inference, exploration is disabled ($\epsilon = 0$), and the agent greedily selects the action with the highest predicted Q-value at each step.

The resulting output is an ordered sequence of locations representing the optimized travel route, which is then returned to the TripMind system.

5 Experiment

5.1 Agent 1

In this section, we describe the training implementation of the foundational model for Agent 1 and the subsequent construction of the vector database used for semantic destination retrieval.

5.1.1 Training setup

The `TripMindEncoder` model was trained using a Multi-task Learning approach to simultaneously perform destination identification and travel category (Intent) prediction.

Experimental Environment and hardware The experimental implementation was partitioned across hai different computing environments to optimize performance:

- Training phase: The model was trained on the Kaggle cloud platform using an NVIDIA Tesla P100 GPU to accelerate the intensive matrix operations of the Transformer's self-attention mechanism.
- Inference and Ingestion Phase: Following weight optimization, the prediction and embedding processes were deployed on an Apple M1 chip. The system utilizes the Metal Performance Shaders (MPS) framework to ensure efficient real-time query responses.

Hyperparameters: The configuration parameters were set to ensure stable convergence and high-fidelity feature extraction, as presented in Table 1:

Table 1: Hyperparameter configuration for Agent 1

Parameter	Value
Total epochs	100
Batch size	32
Learning rate	$5e^{-4}$ (AdamW Optimizer)
Vector dimension (d_{model})	256
Number of Attention Heads	8
Number of Encoder Layers	4
Max sequence length	100 tokens

5.1.2 Training results and visual analysis

The training performance, as recorded in Figure 17, demonstrates the model's high capacity for learning complex semantic structures:

Detailed Curve Analysis: A granular examination of the training dynamics reveals the effectiveness of the selected hyperparameters and architecture:

- Loss Curve (Left): The training loss exhibits a classic exponential decay, starting from approximately 13.5 and converging smoothly toward zero. This behavior indicates that the Learning Rate (5×10^{-5}) is optimally tuned for the loss landscape, preventing significant oscillations or divergence.
- Accuracy Curve (Right): The accuracy progression follows three distinct phases:
  1. Initial Adaptation (Epochs 0-20): Growth is relatively slow as the Transformer architecture learns to interpret spatial relationships through Positional Encoding.
  2. Rapid Learning (Epochs 20-60): The model experiences a surge in performance as it begins to effectively map semantic tokens to destination identifiers.
  3. Fine-tuning (Epochs 80-100): The model enters a final optimization stage, refining its weights to achieve a near-perfect peak training accuracy of 99.44%.

5.2 Agent 2

5.2.1 Dataset and preprocessing

The dataset consists of user-generated reviews collected from social platforms and review websites. Each data sample includes 3 mains fields:

- Review text
- Star rating (1–5)
- Location identifier

For training purposes:

- Reviews with 5-star ratings are labeled as positive
- Reviews with 1-star ratings are labeled as negative

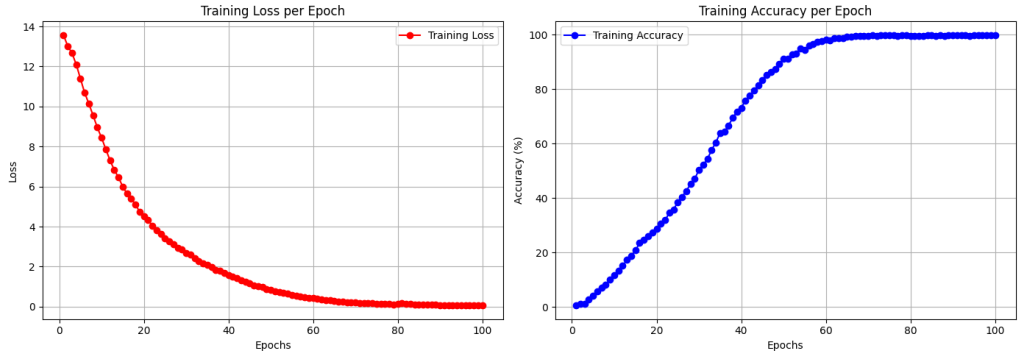

Figure 17: Training process of agent 1

- Reviews with 2–4 stars are treated as unlabeled and later incorporated using pseudo-labeling

Standard text preprocessing techniques are applied, including tokenization, lowercasing, and noise removal.

5.2.2 Results and analysis

Experimental results demonstrate that the model achieves stable and reliable performance on sentiment classification. The attention mechanism significantly improves interpretability by highlighting sentiment-bearing tokens within reviews.

Furthermore, incorporating pseudo-labeled neutral reviews improves coverage without introducing substantial noise, leading to better location-level sentiment estimation.

Table 2: Training and Self-Training Performance of Agent 2 (Sentiment Analysis)

Phase	Epoch	Loss	Accuracy
Initial Training	0	0.1830	0.9710
	1	0.1051	0.9719
	2	0.0643	0.9763
	3	0.0456	0.9798
	4	0.0431	0.9781
	5	0.0426	0.9719
	6	0.0302	0.9737
	7	0.0279	0.9737
	8	0.0214	0.9754
	9	0.0137	0.9728
Self-Training	0	0.0463	0.9907
	1	0.0310	0.9907
	2	0.0236	0.9914
	3	0.0157	0.9907
	4	0.0135	0.9889

5.2.3 Impact on recommendation quality

When integrated into the TripMind pipeline, Agent 2 significantly improves recommendation quality by:

- Reducing poorly reviewed locations
- Promoting consistently positive destinations
- Enhancing user satisfaction in downstream route planning

These results confirm the effectiveness of sentiment-driven ranking as a critical component of the multi-agent recommendation system.

5.3 Agent 3

5.3.1 State representation

Each state encodes the full spatial configuration of the routing problem:

- Latitude and longitude of all five locations.
- A binary visited mask indicating whether each location has been visited.

Formally, the state vector is defined as:

$$s = [lat_1, lng_1, \ldots, lat_5, lng_5, v_1, \ldots, v_5]$$

where $v_i \in \{0, 1\}$ denotes the visited status of location i.

This representation allows the model to reason globally about spatial relationships rather than relying solely on the current position.

5.3.2 Action space and action masking

At each timestep, the agent selects one of five discrete actions, each corresponding to visiting a specific location.

To enforce route validity, action masking is applied:

- Actions corresponding to already visited locations are masked out.

- This prevents invalid transitions and reduces the effective action space.

Action masking significantly improves training efficiency and convergence stability.

5.3.3 Deep Q-Network architecture

The Deep Q-Network consists of:

- Input layer: 15-dimensional state vector. (5 locations $\cdot$ 2 geographical coordinates $+$ 5 binary elements of visited mask)
- Two fully connected hidden layers, each with 128 neurons and ReLU activation.
- Output layer: 5 neurons representing Q-values for each action.

The network is trained to approximate the optimal action-value function over the routing state space.

5.3.4 Reward function

The reward signal is designed to encourage short-distance transitions:

$$r_t = -distance(current\_location, next\_location)$$

This negative distance reward penalizes long moves and implicitly minimizes the total route length. An episode terminates once all five locations have been visited.

5.3.5 Training with multi-input episodes

To enhance generalization, the agent is trained on multiple routing scenarios rather than a single fixed input.

For each training episode:

- A province is selected randomly from dataset.
- Five locations from the same province are sampled.
- A new routing environment is instantiated.
- The agent interacts with the environment to collect experience.

This strategy allows the model to learn reusable routing patterns applicable across diverse geographic configurations.

6 Acknowledgments

This work is supported and supervised by Professor Khoat T. Quang under the course of Introduction of Deep Learning.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other network architectures. *Neural Networks*, 18(5-6):602–610.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30.

Michael Wooldridge and Nicholas R Jennings. 1995. Intelligent agents: theory and practice. *The Knowledge Engineering Review*, 10(2):115–152.