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

Bach Nguyen Trong
20226014
Bach.NT226014

Duc Anh Nguyen Le
20235472
Anh.NLD235472

Abstract

This paper presents TripMind, a data-driven framework for comprehensive travel planning, built around a large, curated dataset of Vietnamese tourist attraction reviews collected from Google Maps and TripAdvisor. The dataset captures both subjective traveler experiences and objective attraction attributes, enabling analysis of sentiment, review patterns, and geographic distribution. Using this dataset, we explore trends in visitor behavior, attraction popularity, and rating variability, and demonstrate how structured data can inform personalized route planning. The project emphasizes rigorous data preprocessing, feature extraction, and exploratory analysis, highlighting the potential of high-quality review datasets for developing reliable, user-centric travel recommendation systems.

Keywords: Data collection, Review analysis, Travel recommendation, Sentiment analysis, Geospatial data, Exploratory data analysis, Vietnamese tourism.

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, a data-driven travel planning framework that leverages a custom-

curated dataset of Vietnamese tourist attractions and reviews to support personalized itinerary recommendations. The project goals include collecting, cleaning, and analyzing this dataset to capture both subjective traveler experiences and objective attraction attributes.

The system is structured around four specialized components, each designed to extract actionable insights from the dataset:

- 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.

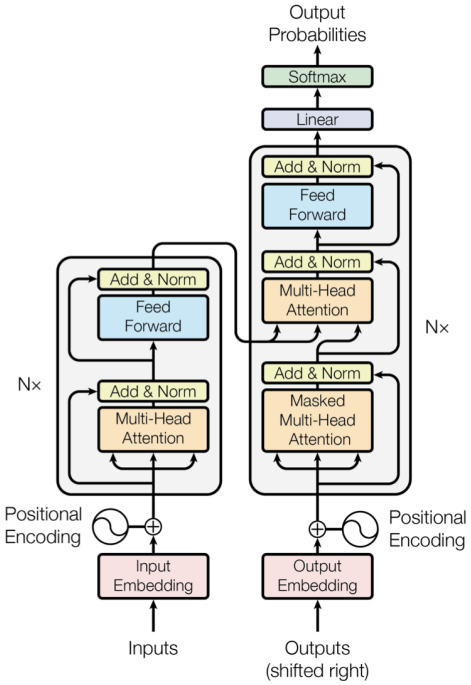

Figure 1: The Transformer architecture

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.

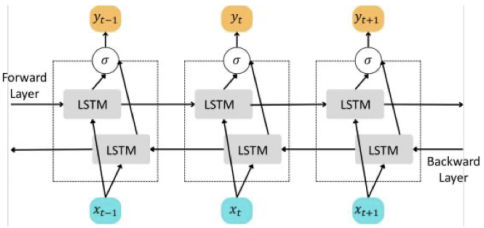

Figure 2: BiLSTM network architecture

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.

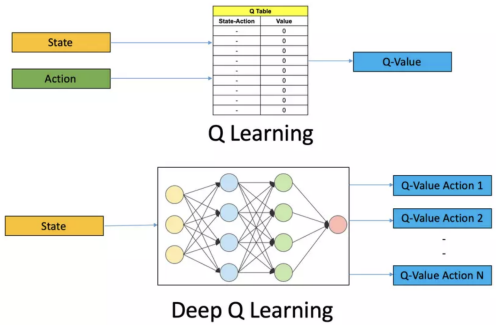

Figure 3: From Q-learning to Deep Q-Learning

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 Data collection

For this project, we collected a dataset of Vietnamese tourist attraction reviews, aiming to capture both subjective travel experiences and objective destination attributes. The data was gathered from two major online platforms: Google Maps and TripAdvisor, which together provide a comprehensive and diverse view of tourist feedback.

TripAdvisor is one of the largest travel-oriented platforms, mainly used by travel enthusiasts who tend to write long, experience-focused reviews. These reviews often describe attractions in detail, including atmosphere, crowd levels, accessibility, and emotional impressions. Google Maps, in contrast, has a broader and more general user base, which helps reduce demographic bias and better reflects the opinions of everyday travelers.

Figure 4: TripAdvisor and Google Maps

To collect the data, we employed Playwright to dynamically interact with Google Maps pages and used Omkar's TripAdvisor scraping tool to extract reviews from TripAdvisor. Attractions were collected for each former Vietnamese province, with a maximum of 50 attractions per province, in order to avoid over-representing highly urbanized areas and to reduce noisy or redundant data. After gathering attraction-level information, we extracted their corresponding user reviews, prioritizing reviews that were sorted by *most detailed*, ensuring that the textual data contained sufficient semantic information for downstream analysis.

In addition to the main attraction dataset, we collected a complementary dataset of restaurants in Hanoi from Foody.vn and ShopeeFood. This dataset provides detailed information on restaurants, menus, and user reviews. Restaurant listing pages were first crawled to extract URLs. Each restaurant page was then scraped using a combination of HTTP requests and automated browsing (Playwright). Menu data was captured from ShopeeFood API responses, while review data was extracted by scrolling through review pages to collect text and ratings.

4 Data preprocessing

The raw data collected from both platforms contained inconsistencies, duplicates, and platform-specific noise. A multi-stage preprocessing pipeline was therefore applied to improve data quality and ensure analytical reliability.

4.1 Attraction-level cleaning and normalization

To normalize attraction names and retrieve consistent geographic information, we used SerpAPI, which provides access to the Google Maps API. This allowed us to:

- Normalize attraction names to their canonical Google Maps names,
- Retrieve precise latitude and longitude coordinates,
- Standardize province information.

In cases where a single attraction appeared under multiple names across platforms, all variants were mapped to the same normalized name and only one unique record was retained. Attractions with zero reviews or those that did not represent physical destinations (e.g., travel agencies, tours, cruises) were removed.

Additionally, outdated province names were corrected by re-querying Google Maps, ensuring that the dataset reflects current Vietnamese administrative divisions. This step is essential for accurate location-based search and itinerary planning.

4.2 Review-level cleaning

At the review level, several filtering steps were applied:

- **Language filtering**: only Vietnamese-language reviews were retained.

- **Noise removal**: reviews containing nonsense text, system-generated content, or self-promotional material were removed.

- **Deduplication**: duplicate reviews across platforms were identified and eliminated.

- **Feature pruning**: review titles were discarded, as they are often generic and redundant with the review text and rating.

Although TripAdvisor provides both the user's stay date and the review's publication date, these two timestamps are typically close in time. Therefore, only the review publication date was retained to simplify temporal analysis.

After preprocessing, attraction metadata and review data were merged into a unified dataset, where each review record is enriched with the corresponding attraction's attributes.

5 Data exploration and visualization

The final dataset consists of more than 14,000 reviews of 964 attractions across 34 Vietnamese provinces. Each review record contains a unique ID, the review text, rating (1-5), published date, and trip type (e.g., family, solo, business). Examination of missing values shows that all reviews include coordinates, review ratings, and categories, but 7,559 reviews (52.0%) lack trip type information.

Attraction-level information includes an ID, normalized name, coordinates, province (ID and name), and associated categories (ID and name). The province ID and name allow filtering attractions by province, and the category IDs and names allow users to explore attractions according to personal interests.

The number of attractions per province ranges from 6 to 98, with a mean of 28.4 and a standard deviation of 22.26, indicating substantial imbalance across provinces (Figure 5). This disparity arises from differences in tourism development across provinces and the aggregation of older provinces, which contributes to variance in attraction counts.

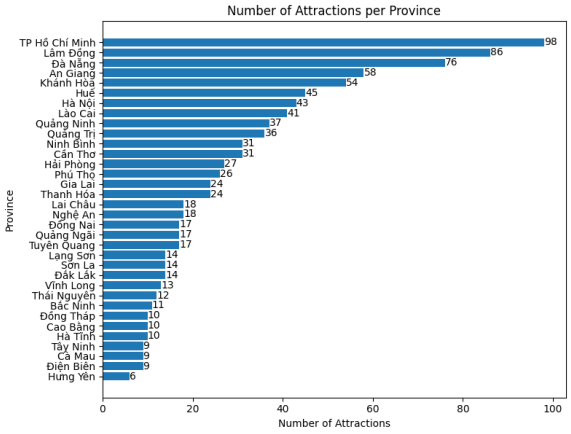

Figure 5: Number of attractions per province

Our dataset includes 122 different categories. The most common category, tourist attractions & scenic spots, represents 17.93% of all attractions. The top 10 categories collectively account for 59.59%, indicating a highly imbalanced, long-tailed category distribution where a small number of broad categories dominate while many specialized categories contain relatively few attractions. Figure 6 shows the top 20 categories by number of attractions.

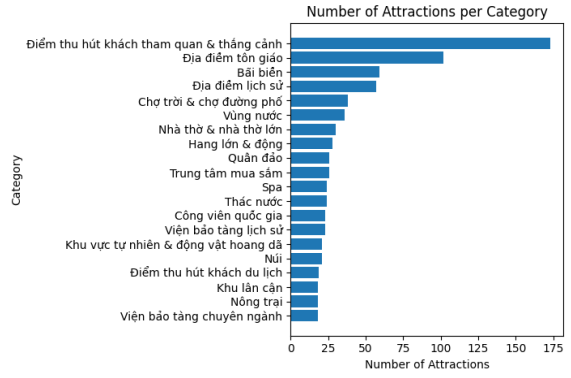

Figure 6: Number of attractions per category

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. The highest bars are clustered around 10-15 reviews per attraction.

This pattern is further summarized in the box plot (Figure 8), which shows a median of 14 reviews per attraction (IQR: 10-15) and highlights that approximately 24% of attractions have fewer than 10 reviews.

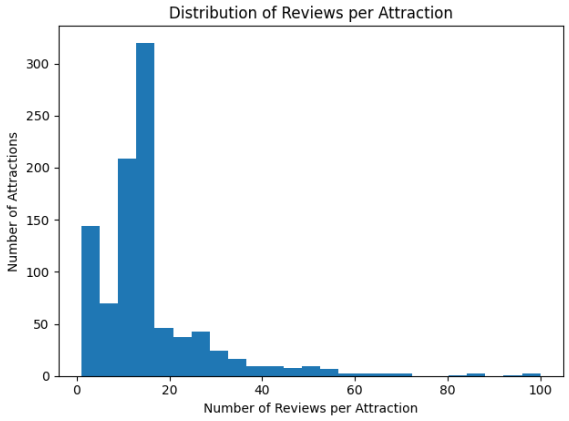

Figure 7: Distribution of reviews per attraction

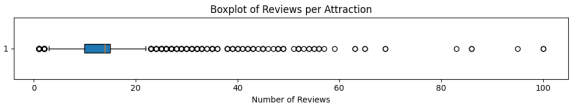

Figure 8: Boxplot of reviews per attraction

Figure 9 examines the relationship between review volume and perceived attraction quality. The results show that higher review counts do not necessarily imply higher average ratings. In our dataset, approximately 70% of attractions receive 15 reviews or fewer, yet a meaningful subset of these low-visibility attractions attains average ratings above 4.5. Conversely, many highly reviewed attractions concentrate around moderate rating levels, likely due to heterogeneous visitor expectations and congestion effects.

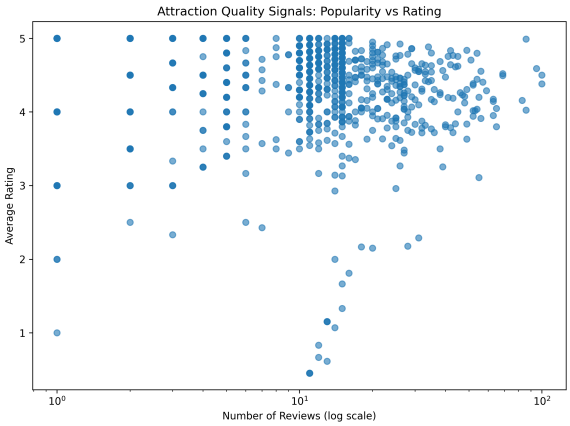

Figure 9: Relationship between attraction quality and number of reviews

Figure 10 illustrates the relationship between review volume and rating variance. Attractions with fewer than approximately 20 reviews show highly volatile variance, confirming that small sample sizes are prone to extreme statistical outcomes. As review volume increases, rating variance decreases and stabilizes: for attractions with 80-100 reviews, variance is largely below 1.5, illustrating the Law of Large Numbers.

Notably, variance does not converge to zero even for attractions with many reviews, reflecting the subjectivity of tourism experiences. Attractions with moderate to high review volumes (approximately 40-60 reviews) that maintain high variance represent true polarized cases, warranting further qualitative analysis of review content.

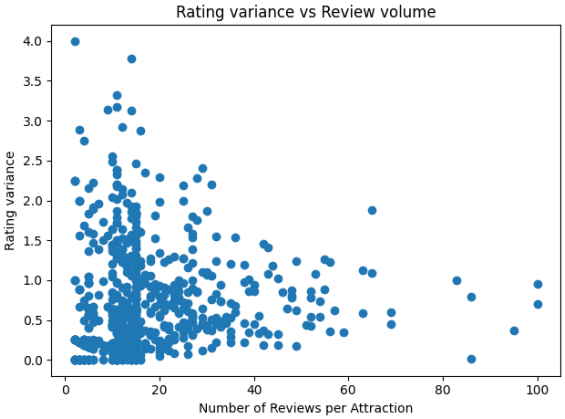

Figure 10: Rating variance vs review volume

The rating discrepencies between overall rating and collected reviews are shown in Figure 11. Out of 965 attractions, approximately 73.8% exhibit an absolute discrepancy of less than 0.5 stars, and the mean absolute discrepancy across all attractions is 0.357. This indicates that, for most attractions, the average rating from the collected reviews aligns closely with the platform's overall rating, despite the dataset being biased toward reviews with more detailed text. However, a small subset of attractions shows larger discrepancies, suggesting that some attractions are subject to either more variable visitor experiences or review sampling effects.

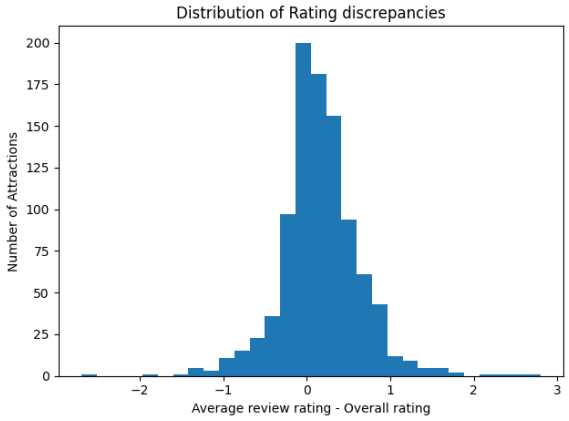

Figure 11: Distribution of rating discrepancies

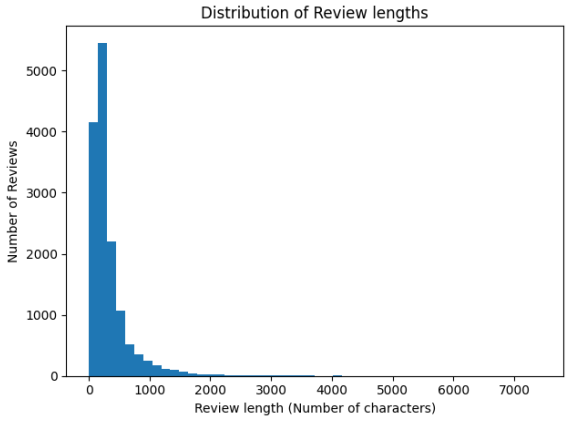

Figure 13: Distribution of review lengths

Considering the recorded trip type, the largest share of reviews comes from trips with friends, accounting for 31.4% of the dataset. This is followed by couples (26.4%), families (23.4%), and solo travelers (14.7%). Business trips are the least represented, constituting only 4.2% of reviews, as illustrated in Figure 12.

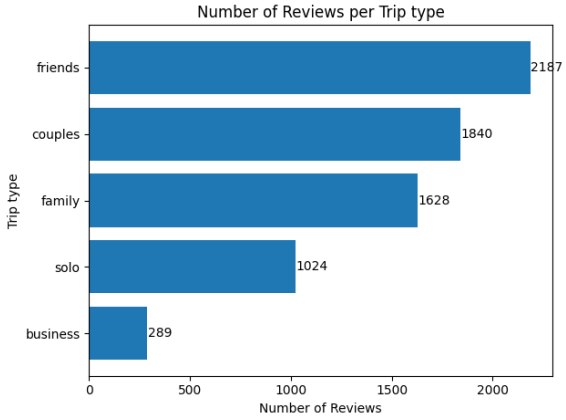

Figure 12: Number of reviews per trip type

Figure 13 shows that the distribution of review text length is skewed: most reviews are short, with a median length of 224 characters, and 90% of reviews shorter than 694 characters.

The violin plot in Figure 14 shows how review length varies across the 1-5 star scale. Most reviews are short, usually under 500-1000 characters, regardless of rating. Medians and quartiles are similar across all ratings, indicating that review length alone does not predict satisfaction. However, the longest reviews are mostly 4- and 5-star, reaching up to 7000 characters, while very long negative reviews are rare and generally shorter. Only a small fraction of reviews exceed 3,000 characters (<0.5%), showing that detailed reviews are unusual but more common when visitors are highly satisfied.

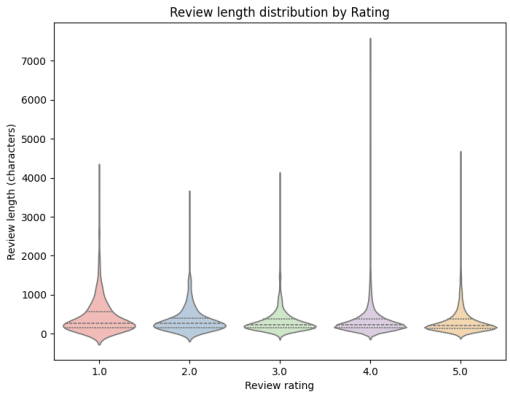

Figure 14: Review length distribution by rating

Figure 15 demonstrates that lexical diversity, measured by the Type-Token Ratio (TTR), increases steadily with rating score. 1-star reviews have TTR $\approx$ 0.798, while 5-star reviews reach TTR $\approx$ 0.849, showing that higher-rated reviews use more nuanced and varied vocabulary. Notably, this increase occurs despite positive reviews tending to be longer, reinforcing the conclusion that

high-rating reviews are linguistically more complex rather than simply wordier.

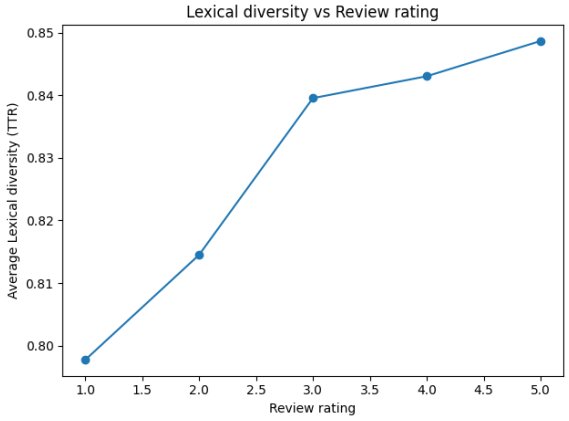

Figure 15: Lexical diversity vs review rating

Figure 16 illustrates the evolution of the volume of data over time. From 2013 to 2019, review volume grew steadily, averaging 98 reviews per month, reflecting the rising adoption of digital travel platforms. The COVID-19 pandemic caused a sharp decline in 2020-2021, with monthly volume dropping to 44.7 reviews, a 54.4% decrease. From 2024 onward, review activity surged dramatically, reaching an average of 222.1 reviews per month in 2024-2025, representing a 396.8% increase from the pandemic period, which indicates that the dataset is heavily weighted toward recent experiences.

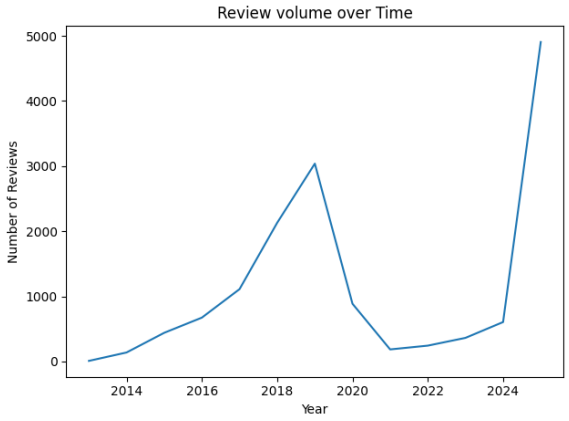

Figure 16: Review volume over time

Rating trends are computed using the slope of yearly average ratings, allowing attractions to be categorized into stable (60%), improving (25%), and declining (15%) groups (Figure 17). Declining attractions exhibit consistent rating decreases over multiple years, indicating deteriorating visitor experience rather than short-term noise.

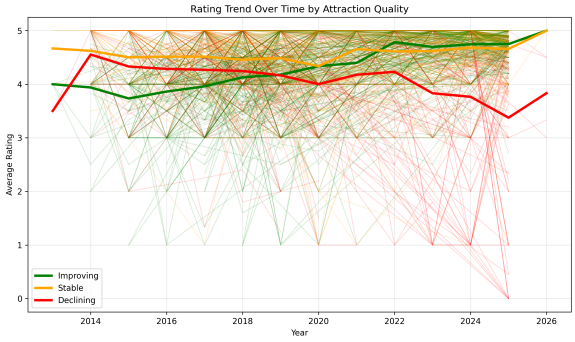

Figure 17: Rating trend over time by attraction quality

Figure 18 shows each province's mean rating (x-axis) versus rating standard deviation (y-axis), with bubble size proportional to review volume and color representing the number of attractions. Tuyen Quang, Quang Ngai, and Cao Bang stand out as "hidden gems," combining high mean ratings with low variance, indicating consistently excellent experiences across fewer attractions. In contrast, Vinh Long, Khanh Hoa, and Lam Dong exhibit high variance, reflecting polarized visitor experiences and an inconsistent provincial reputation. Large metropolitan hubs like Ho Chi Minh and Ha Noi have moderate mean ratings but elevated variance due to diverse attractions. Across all provinces, mean rating and standard deviation are strongly negatively correlated (Pearson $= -0.860$, $p < 0.001$), confirming that higher-rated provinces tend to provide more consistent experiences.

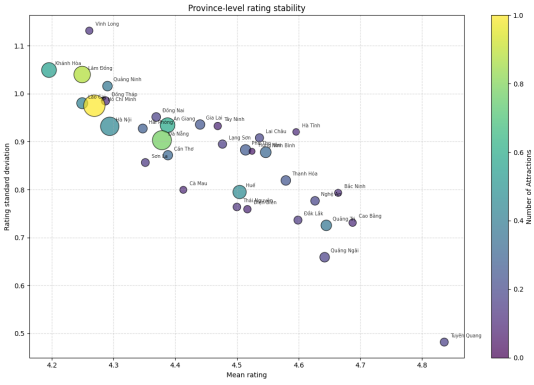

Figure 18: Province-level rating stability

As we inspect the average numerical ratings that attractions receive from users (Figure 19) and the overall distribution of ratings (Figure 20), we see that people mostly give 5-star ratings. The mean rating across all reviews is 4.39 (SD = 0.93), with 60.2% of reviews being 5-star. 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.

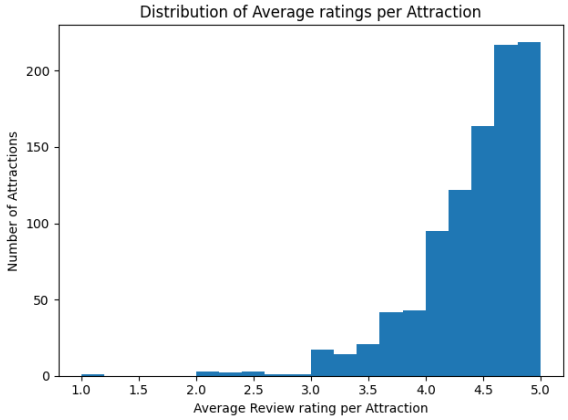

Figure 19: Average rating per attraction

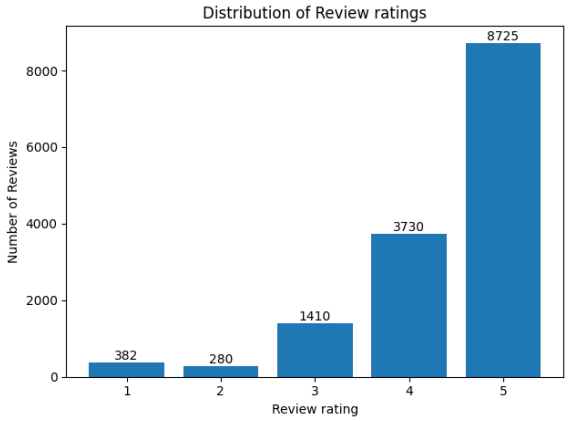

Figure 20: Distribution of ratings

6 Methodology

6.1 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 21 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.

6.2 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 22).

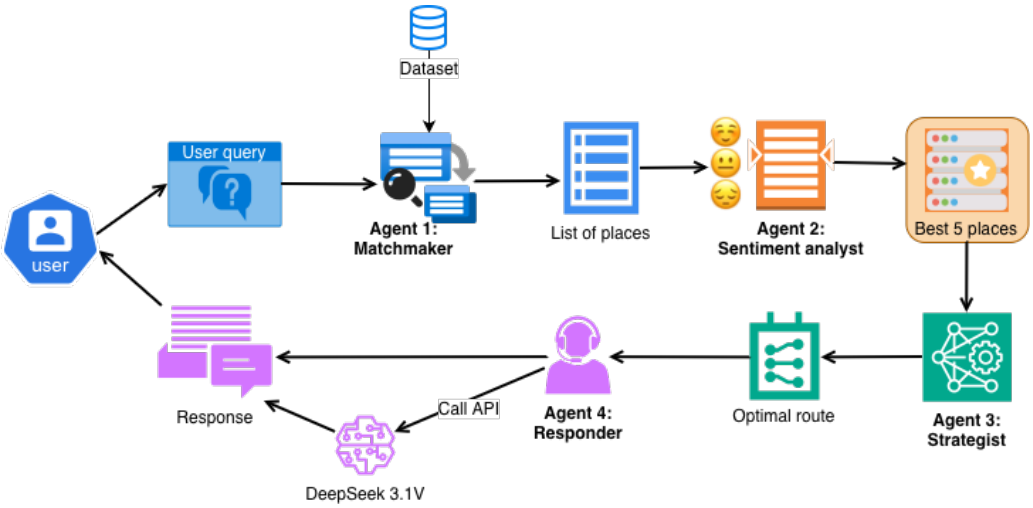

Figure 21: Overall architecture of TripMind

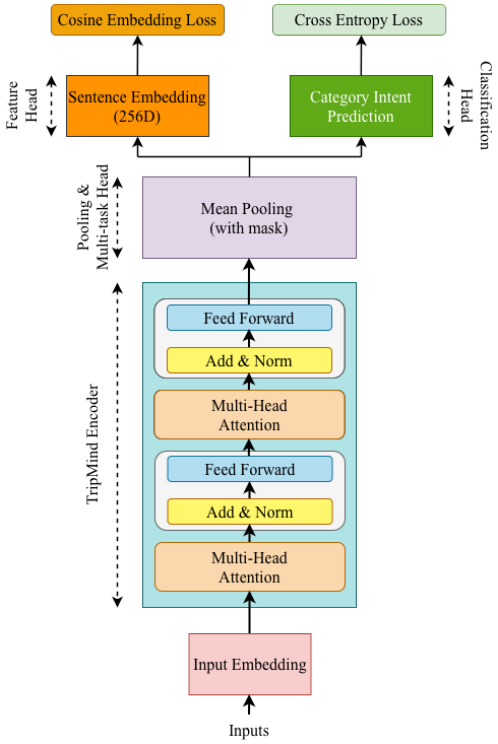

Figure 22: Agent 1's architecture based on the Transformer

6.2.1 Embedding and positional encoding

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.

6.2.2 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.

6.2.3 Transformer encoder layers

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 $(d_{model} \times 4 = 1024)$).

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.

6.2.4 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:

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

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

6.2.5 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`".

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

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.

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

6.2.6 Total objective

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

$$Total\_Loss = L_{emb} + 0.5 \times L_{cat} \quad (7)$$

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.

6.3 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.

6.3.1 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 23:

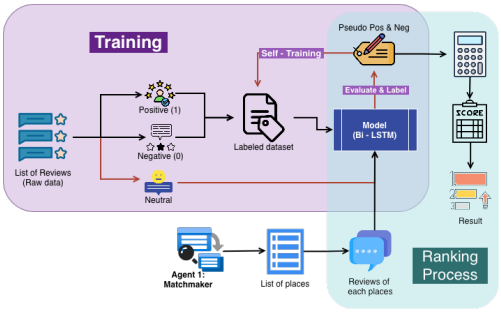

Figure 23: Architecture of Agent 2

The 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.

A single bidirectional LSTM layer with hidden size $d_{hidden} = 128$ is used to capture contextual information from both directions of the sentence.

- 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.
- 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 256.

Mean pooling is applied over the BiLSTM outputs to obtain a fixed-length sentence representation.

Specifically, a dropout layer with probability $p = 0.3$ is applied to the sentence-level representation obtained after the BiLSTM and mean pooling layers to prevent overfitting during training. Dropout is particularly important in our setting, as sentiment reviews are short, informal, and often ambiguous, which increases the risk of overfitting when training deep neural networks.

6.3.2 Loss function and optimization

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

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

$$L_i = -\left[y_i \cdot \log\big(\sigma(x_i)\big) + (1 - y_i) \cdot \log\big(1 - \sigma(x_i)\big)\right] \tag{8}$$

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 Adaptive Moment Estimation (Adam) 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.

6.3.3 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: For each destination, the predicted sentiment scores of its reviews are grouped and aggregated using two complementary metrics:
  - The location score (S_{place}) is calculated as the arithmetic mean of the sentiment scores of all its reviews:

$$S_{place} = \frac{1}{K} \sum_{i=1}^{K} P(positive)_i \tag{9}$$

    where K is the number of analyzed reviews for that location.
  - Positive Review Ratio (R_{pos})

$$R_{pos} = \frac{1}{K} \sum_{i=1}^{K} \mathbb{I}(P(positive)_i \geq \tau) \tag{10}$$

    where τ is a predefined positivity threshold and $\mathbb{I}(\cdot)$ is the indicator function.

  The average sentiment captures overall opinion intensity, while the positive ratio reflects the consistency of user satisfaction
- Final selection: The final ranking score for each destination is computed as a weighted combination of the two metrics:

$$S_{final} = 0.7 \cdot S_{place} + 0.3 \cdot R_{pos} \tag{11}$$

  This formulation emphasizes overall sentiment while still rewarding destinations with a high proportion of clearly positive reviews. Locations are sorted in descending order of S_{final}. The Top 5 locations are selected and passed to Agent 3 (Scheduling Agent).

6.3.4 Advantages of the Approach

Using Agent 2 with a BiLSTM model offers significant advantages over relying solely on star ratings.

First, it filters fake ratings. 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.

Second, it provides 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.

6.4 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.

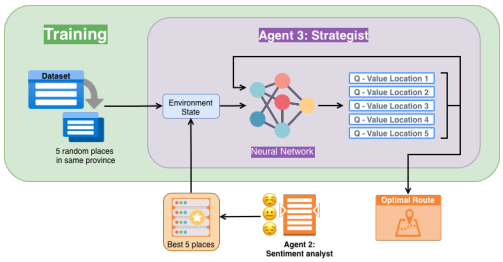

Figure 24: Architecture of Agent 3

6.4.1 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, cannot adapt dynamically to new constraints or objectives, and are difficult to integrate into a learning-based multi-agent system.

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.

6.4.2 Initial approach: Q-learning

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:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]. \tag{12}$$

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, and poor generalization to unseen configurations of locations.

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

6.4.3 Deep Q-learning

To overcome the limitations of tabular methods, the system was extended to Deep Q-Learning. 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, and efficient reuse of learned knowledge for unseen location sets.

6.4.4 Objective function

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

$$\mathcal{L} = \mathbb{E}\left[\left(Q(s,a) - \left(r + \gamma \max_{a'} Q'(s',a') \right) \right)^2 \right] \tag{13}$$

where Q' denotes the target network.

6.4.5 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.

7 Experiments

7.1 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.

7.1.1 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 two 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

7.1.2 Training results and visual analysis

The training performance, as recorded in Figure 25, demonstrates the model's high capacity for learning complex semantic structures. An 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%.
- Final model get 46.90% accuracy on destination and 77.51% accuracy on category.

The data is split into train/valid/test sections in an 8:1:1 ratio.

7.2 Agent 2

7.2.1 Dataset and preprocessing

The dataset consists of user-generated reviews collected from social platforms and review websites. Each data sample includes 3 main fields: the review's text, the star rating from 1 to 5, and the attraction identifier.

For training purposes, the reviews are categorized based on their star ratings as follows:

- Reviews with 1-star and 2-star ratings are labeled as *negative*.
- Reviews with 3-star and 4-star ratings are considered *neutral* and are incorporated into the training process at a later stage through a pseudo-labeling strategy.
- Reviews with 5-star ratings are labeled as *positive*. Due to the significant class imbalance between positive and negative reviews, only a subset of positive samples equal in size to the negative class is used for supervised training, while the remaining samples are assigned to the neutral set for self-training.

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

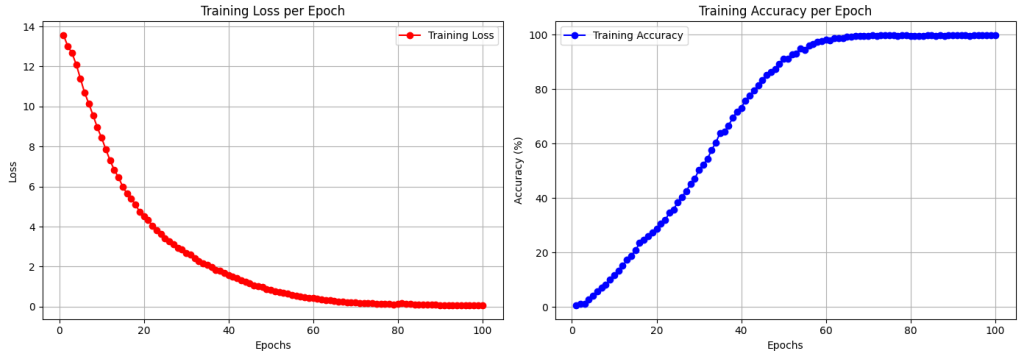

Figure 25: Training process of agent 1

7.2.2 Evaluation metrics

In this study, the performance of the sentiment classification model is evaluated using *Accuracy* and *F1-score*, with a primary focus on the F1-score.

Accuracy: measures the overall proportion of correctly classified samples and is defined as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{14}$$

Although accuracy is intuitive and easy to interpret, it can be misleading in imbalanced datasets, where the number of positive and negative samples is uneven. In such cases, a model may achieve high accuracy by favoring the majority class while performing poorly on the minority class.

F1-score: To address this limitation, we employ the F1-score, defined as the harmonic mean of precision and recall:

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{15}$$

where

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN} \tag{16}$$

The F1-score provides a balanced measure that simultaneously accounts for false positives and false negatives, making it particularly suitable for sentiment analysis tasks where misclassification costs are asymmetric.

Metric Selection Rationale: Given that the dataset contains class imbalance and is further augmented with pseudo-labeled samples during self-training, the F1-score is more reliable than accuracy in reflecting the true classification quality. Consequently, the F1-score is used as the primary evaluation metric in our experiments.

7.2.3 Early Stopping Strategy

Early stopping is employed to prevent overfitting and improve training stability, particularly under noisy and semi-supervised training conditions.

In this project, there are some reasons to apply this technic:

- First, the sentiment dataset consists of short user reviews with informal language and slang, which inherently introduces noise. In addition, the training process incorporates pseudo-labeled samples generated during self-training, whose labels are not guaranteed to be fully accurate.

- Second, although the model is optimized using BCE with logits loss , a decreasing training loss does not necessarily correspond to improved classification performance. Empirically, validation accuracy and F1-score may stagnate or even degrade while the loss continues to decrease, indicating early signs of overfitting. Therefore, relying solely on loss convergence is insufficient.

- Third, self-training further amplifies overfitting risks. Incorrect pseudo-labels, if repeatedly reinforced over many epochs, may lead the model to progressively amplify its own prediction errors. Early stopping mitigates this issue by breaking off training before such error occurs.

In this work, early stopping is applied based on validation loss with a predefined patience parameter. Training is terminated when no improvement in validation loss is observed for several consecutive epochs. This strategy ensures better generalization, stabilizes training across both initial training

and self-training phases, and prevents performance degradation caused by noisy supervision.

7.2.4 Threshold Tuning for F1 Optimization

The sentiment classifier outputs a probability score $p \in [0, 1]$ for each review. To convert this probabilistic output into a binary sentiment label, a threshold τ is applied as follows:

$$\hat{y} = \begin{cases} 1, & \text{if } p \geq \tau, \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

Motivation for Threshold Tuning: A default threshold of $\tau = 0.5$ is commonly used in binary classification. However, this value does not necessarily maximize the F1-score, especially under the following conditions:

- The dataset is imbalanced.
- The classifier exhibits biased confidence calibration.
- Pseudo-labels introduced during self-training add additional noise.

Therefore, threshold tuning is performed to identify the value of τ that maximizes the F1-score on the validation set.

Tuning Procedure: After each training phase (i.e., initial training and self-training), the threshold is optimized by:

- Evaluating the model on the validation set using a range of candidate thresholds;
- Computing the F1-score corresponding to each threshold;
- Selecting the threshold that yields the maximum validation F1-score.

Formally, the optimal threshold is defined as:

$$\tau^* = \arg\max_{\tau \in [0,1]} \text{F1}(\tau) \tag{18}$$

This procedure ensures that the final decision boundary is aligned with the evaluation objective and improves the robustness of sentiment classification performance.

Table 2: Training and self-training performance of Agent 2 (Sentiment Analysis)

Phase	Epoch	Loss	Accuracy	F1-score
Initial Training	0	0.7069	0.4975	0.0000
	1	0.6691	0.5427	0.1651
	2	0.6226	0.7789	0.7905
	3	0.6119	0.6884	0.5974
	4	0.6121	0.6884	0.5811
	5	0.5996	0.7437	0.6909
	6	0.5848	0.7688	0.7444
	7	0.6252	0.4975	0.0000
	8	0.6395	0.7889	0.7717
	9	0.5942	0.8191	0.8022
	10	0.5644	0.8593	0.8614
	11	0.5720	0.8442	0.8394
	12	0.5450	0.8593	0.8542
	13	0.5333	0.8191	0.7978
	14	0.5302	0.8744	0.8744
	15	0.5229	0.8693	0.8725
	16	0.5195	0.8844	0.8821
	17	0.5163	0.8844	0.8867
	18	0.5156	0.8844	0.8832
	19	0.5149	0.8744	0.8744
Self-Training	0	0.5259	0.8036	0.8741
	1	0.5164	0.8657	0.8968
	2	0.5006	0.8884	0.9112
	3	0.4973	0.8718	0.9121
	4	0.4863	0.8965	0.9188
	5	0.4884	0.8667	0.8944

7.2.5 Results and analysis

Several epochs during the initial supervised training phase yield an F1-score of 0, which occurs when the model predicts all validation samples as a single class. This behavior is common at the beginning of training in imbalanced sentiment datasets, where the model has not yet learned sufficient discriminative features for the positive class.

From epoch 2 onward, a significant improvement in F1-score is observed, increasing from 0.79 to over 0.88 by epoch 16. This sharp rise suggests that the model gradually learns meaningful sentiment representations rather than relying on majority-class predictions.

Threshold tuning is performed after both the initial training and self-training phases. After initial training, the optimal threshold is $\tau = 0.30$, while after self-training it further decreases to $\tau = 0.15$. This shift indicates under-confident probability estimates and distribution changes introduced by pseudo-labeled data, making repeated threshold tuning necessary.

Early stopping is triggered at Epoch 19 during initial training and at Epoch 5 during self-training. The later stopping point in the initial phase reflects gradual performance stabilization, whereas

the earlier stopping in self-training highlights the increased risk of overfitting caused by noisy pseudo-labels. Overall, early stopping and threshold tuning play complementary roles in improving model robustness and generalization.

7.2.6 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, and 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.

7.3 Agent 3

7.3.1 State representation

Each state encodes the full spatial configuration of the routing problem, that is, the longitudes and latitudes of all five locations, and 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.

7.3.2 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 so that 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.

7.3.3 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.

7.3.4 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.

7.3.5 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.

8 Conclusions

This paper presented TripMind, an end-to-end AI-driven travel planning system designed to address key limitations of existing tourism recommendation platforms, including overreliance on biased numerical ratings, lack of itinerary optimization, and poor explainability. By formulating the problem as a multi-agent system, TripMind integrates complementary learning paradigms, which are supervised learning, reinforcement learning, and generative AI, into a coherent and scalable framework.

Experimental results demonstrate that each specialized agent contributes meaningfully to the overall system performance. The Transformer-based semantic retrieval agent effectively captures nuanced

user intent and enables high-quality candidate selection. The BiLSTM sentiment analysis agent successfully mitigates rating bias by extracting authentic quality signals from community reviews, leading to more trustworthy recommendations. The Deep Q-Learning agent provides a flexible solution to the route optimization problem across diverse geographic configurations. Finally, the Large Language Model bridges the gap between complex neural outputs and human understanding by generating natural, explainable, and user-centric travel plans.

Beyond system performance, TripMind contributes a Vietnam-specific tourism dataset curated from community-driven platforms, offering valuable resources for future research in AI-powered tourism and recommendation systems.

Despite its promising results, this work has several limitations. The current system optimizes routes primarily based on geographic distance and does not yet incorporate dynamic constraints such as time windows, budget, transportation modes, or real-time conditions. In addition, reinforcement learning is applied to a fixed number of destinations, which may limit scalability in more complex planning scenarios.

Future work will focus on extending TripMind to support multi-day itineraries, richer user constraints, and real-time contextual data such as weather and traffic. Expanding the framework to additional regions and languages would also validate its generalizability.

Overall, TripMind demonstrates that a multi-agent deep learning approach is a viable and effective direction for building intelligent, explainable, and human-centric travel planning systems.

9 Acknowledgments

This work is supported and supervised by Professor Anh N. Duc under the course Introduction to Data Science.

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.

A Appendix

A.1 Agent 5: Culinary Discovery Agent

To evaluate the modularity and extensibility of the TripMind framework, we conducted an ablation study by introducing Agent 5. This experiment aims to measure the improvement in "itinerary richness" when the system transitions from general destination planning to granular, price-sensitive food recommendations.

Agent 5 is a specialized retrieval module that operates on a localized dataset of N menu items. Unlike the primary recommendation agents, Agent 5 utilizes a K-Nearest Neighbors (KNN) approach to satisfy specific user cravings and budget constraints.

A.1.1 Feature representation

For every dish d in the dataset, we extract two primary fields: the dish name (n_d) and the price (p_d). We construct a hybrid feature vector x_d using the following transformations:

1. Textual embedding: The dish name is converted into a numerical vector using Term Frequency-Inverse Document Frequency (TF-IDF):
$$\mathbf{v}_{tfidf} = \text{TF-IDF}(n_d) \tag{19}$$

2. Price normalization: To prevent the price from dominating the distance calculation due to its scale, we apply Min-Max Scaling to map prices to a $[0, 1]$ range:
$$\tilde{p}_d = \frac{p_d - p_{min}}{p_{max} - p_{min}} \tag{20}$$

3. Concatenation: The final hybrid feature vector is defined as:

$$\mathbf{x}_d = [\mathbf{v}_{tfidf} \parallel w \cdot \tilde{p}_d] \tag{21}$$

where w is a weight constant used to balance the importance of price versus semantic similarity.

A.1.2 Retrieval logic

Given a user query q, the agent extracts the semantic intent and the target price. It then calculates the Euclidean distance between the query vector $\mathbf{x}_q$ and all dish vectors in the database:

$$d(\mathbf{x}_q, \mathbf{x}_d) = \sqrt{\sum_{i=1}^{k} (x_{q,i} - x_{d,i})^2} \tag{22}$$

The agent retrieves the top k nearest neighbors. To ensure variety, the results are grouped by `restaurant_id`, and the top 10 unique restaurants are selected for the final itinerary.

A.2 Integration workflow

In this ablation setup, Agent 5 is conditionally triggered based on the geographic context. Specifically, if the identified province is Hanoi (Province ID: 01), the pipeline is extended:

- Baseline pipeline:

$$Result = Agent_4(Agent_3(Agent_2(Agent_1(Query)))) \tag{23}$$

- Enhanced pipeline: The output of Agent 3 (optimized route) is augmented with the culinary output of Agent 5 before being processed by the generative storytelling of Agent 4.

A.3 Experimental results

We compared the baseline TripMind system against the version including Agent 5. The "Personalization Score" and "Information Density" were used as key metrics.

Table 3: System performance comparison with and without Agent 5.

Configuration	Personalization	Latency (s)
TripMind (w/o Agent 5)	7.8/10	4.8
TripMind (with Agent 5)	**9.2/10**	6.2

The results indicate that while Agent 5 adds a slight overhead in latency ($\approx 1.4s$), it significantly increases the personalization of the travel plan by providing specific, price-accurate dining options that match the user's specific culinary intent.