electronics

MDPI

Article

Improving Location Recommendations Based on LBSN Data Through Data Preprocessing

Robert Bembenik *, Mateusz Orzoł and Piotr Maciąg

Institute of Computer Science, Warsaw University of Technology, Nowowiejska 15/19, 02-665 Warsaw, Poland
* Correspondence: robert.bembenik@pw.edu.pl

Abstract: The accurate prediction of the next location in a sequence is highly beneficial for users of mobile applications. In this study, we investigate how various data preprocessing techniques affect the performance of location recommendation systems. We utilize datasets from Foursquare and Twitter, incorporating users' historical check-ins. Key preprocessing steps include filtering datasets to users with common features, analyzing user location preferences, varying sequence lengths and location categories, and integrating time-of-day information. Our findings reveal that proper data preprocessing significantly enhances the accuracy of recommendations by addressing key challenges such as data sparsity and user heterogeneity. Specifically, tailoring datasets to individual user attributes improves model personalization, while restructuring category hierarchies balances precision and diversity in the recommendations that are given. Integrating temporal data further refines the predictions that are made by accounting for time-based user behavior. Recommendations are generated using recurrent neural networks (RNNs) and hidden Markov models (HMMs), with the experimental results showing up to 20% improvement in the precision of personalized models compared to global ones.

Keywords: location recommendation; data preprocessing; recurrent neural networks; hidden Markov models

Academic Editor: Domenico Ursino

Received: 29 November 2024
Revised: 20 January 2025
Accepted: 29 January 2025
Published: 11 February 2025

Citation: Bembenik, R.; Orzoł, M.; Maciąg, P. Improving Location Recommendations Based on LBSN Data Through Data Preprocessing. *Electronics* **2025**, *14*, 701. https://doi.org/10.3390/electronics14040701

1. Introduction

Recommending a place to visit to a user of a recommendation system is not an easy task. Contributing to this difficulty are, among others, these factors:

- Data incompleteness: Many approaches to recommending the next place to visit to a user that are offered in the literature are based on data from location-based social networks (LBSNs). The specificity of LBSN data relies on users marking their whereabouts (check-ins) in such networks, which they do only when they feel like doing so. For example, they oftentimes check in to fancy places, or places that are for some reason particularly interesting to them. The reason for this is usually to let their friends on the social network know of the extraordinary places they visit. To this end, only a fraction of the places that they visit are registered on LBSNs. From the perspective of using such a dataset to recommend a next place to visit to a user, the available information is based on mostly special places in some way, and does not include a vast majority of the places the user actually visited during the day. The data are thus incomplete from the perspective of using them as a basis for generating recommendations;
- Weather: Weather can be a very important factor when deciding to choose one place to visit over another. For example, if we were looking for a recommendation of a

place to visit in the afternoon, our choices would be potentially different if the weather was warm and dry, and they would be different still if it was raining (e.g., we would probably prefer to go to an open air place in the first case and an indoor place in the second);

- Company: The choice of a place to visit can be influenced by the company we are in. If we are alone, the choice might be different than when we are in a group of people (e.g., family or friends). In the second case, other people will influence our choices, so these choices will be different than if we were choosing the place ourselves;
- Varying interests over time: People's habits and interests tend to change slightly over time. Due to these changes, not all the places that we visited, e.g., 2 years ago will still arouse our interest at the present moment;
- Multi-level periodicity [1]: Some places people tend to visit quite regularly, e.g., on a daily or weekly basis (like going to work, to a gym, etc.). Some, however, are visited less often, but are still visited with some regularity as well. An example might be a visit to a dentist, to a music festival, etc.

Not taking into account the above circumstances as a whole significantly reduces the accuracy of location recommendation, even if the location recommendation algorithm itself is quite efficient. However, in this paper, we do not consider the influence of the abovementioned factors on the recommendation precision, as that would require dedicated methods that careful incorporating these factors in the final predictions to be produced.

Our focus here is on examining how different data preprocessing strategies impact the accuracy of location recommendations generated using popular methods, namely neural networks and hidden Markov models. Specifically, we explore the effects of considering only users with shared features or common locations, varying the sequence length of visited places, altering the number of categories for these locations, and incorporating time-of-day information. The results of these examinations can be especially interesting from the business perspective, i.e., they can help improve the efficiency of location-based services, enhance user engagement through personalized recommendations, and support targeted marketing strategies by allowing for a better understanding of user behavior and preferences.

To achieve this, we implement and evaluate several transformations on LBSN data to assess their impact on the precision of location recommendations. The implemented transformations are as follows:

- Narrowing the input dataset to users sharing some common features;
- Utilizing user similarity in location preferences based on users' history of visited locations;
- Adjusting the length of the sequences considered;
- Varying the number of location categories;
- Incorporating time-of-day information in the recommendation computation.

The rest of this paper is organized in the following way. Section 2 reviews related work in location-based recommendation systems, categorizing existing approaches such as collaborative filtering, Markov models, and neural networks. Section 3 describes the preprocessing methods applied to the datasets, including the sequence generation, the restructuring of location categories, and the incorporation of temporal data. Section 4 explains the implementation and configuration of the tested recommendation systems, focusing on recurrent neural networks (RNNs) and hidden Markov models (HMMs). Section 5 gives data characteristics. Section 6 discusses the computational costs and scalability of the proposed solutions. Section 7 presents the experimental results, evaluating the impacts of the data preprocessing strategies on the accuracy of the resulting recommendations. It also provides a detailed comparison of the models based on their performance. Section 8 concludes the paper, summarizing the findings and proposing directions for future research

to further improve location-based recommendation systems. All abbreviations used in the paper are explained in section Abbreviations.

2. Related Work

In the literature, one can find multiple approaches to solving the task of next-location recommendation. These approaches can be categorized, subjectively, into several groups. We present an overview of the existing methods, divided into the following categories: collaborative filtering and matrix factorization, Markov models, neural networks, and other methods.

The first group of methods, using *collaborative filtering and matrix factorization*, uses an approach stemming from the one used in recommending regular items, adjusting it to the specificity of the spatial context. In [2], the authors propose a method of recommendation based on collaborative filtering that additionally incorporates Ebbinghaus's memory theory [3–5], taking into consideration people's natural tendency to forget things over time. In [6], the authors apply collaborative filtering to learn users' transition patterns between location categories using other users' similar transition patterns. Similar users are clustered based on their check-in frequency in different place categories. Matrix factorization is used for each cluster to predict preference transitions. In [7], the authors use an observation that individual visiting locations tend to cluster together. They reflect this observation in the proposed factorization model. In the proposed system, a weighted matrix factorization approach is used. In particular, users' latent factors are augmented with activity area vectors and points of interests' (POIs') latent factors are augmented with influence area vectors. Reference [8] considers rankings of factorizations. In the proposed solution, it is assumed that POIs with a higher number of check-ins are of higher interest to users. Users' preference rankings for POIs are fitted to learn the latent factors of users and POIs. The solution also takes into account the contribution of unvisited POIs. Reference [9] proposes a factorization model using multiple feature spaces that is capable of using multiple context types in POI recommendation. In particular, the approach improves the method given in [8] by removing the interaction factor matrix and splitting the POI latent space into slices with different context information. Reference [10] aims to recommend new places to users, ones they have not previously visited. The proposed approach consists of two steps. Firstly, a set of potential locations is learned from three types of friends (social friends, location friends, and neighboring friends). Next, the learned potential locations of each user are incorporated into a matrix factorization model with different error loss functions. The generated recommendation strategies cover standard recommendation, location cold-start recommendation, and user cold-start recommendation.

The next group of methods of next-location recommendation uses *Markov chains*. This approach to recommendation characterizes, especially, early works in the field. Reference [11] uses a mobility Markov chain to predict the next place a user will visit. As opposed to other methods presented in this section, the tests conducted by the authors did not include LBSN data but instead used GPS traces of (mostly) researchers. The presented method consists of two steps. Firstly, POIs are identified via a clustering algorithm. Next, transitions and relative probabilities are computed, whereas their chronological order, obtained from mobility traces representing POIs, is preserved. Lastly, the transitions between states, taking into account the n last visited states, are computed. Reference [12] predicts the next locations for pedestrian movement. The locations are represented by their timestamp, longitude, and latitude. Firstly, the locations are clustered based on the temporal information, i.e., daytime, nighttime, and weekend events. Next, the created clusters are used to train different hidden Markov models that correspond to the different types of location histories. A new sequence of visited locations is first assigned to the appropriate cluster.

Then, inference is conducted using the corresponding HMM to discover the most probable next location. Locations are approximated here by a decomposition of the Earth's surface into triangular meshes of variable resolutions. Reference [13] extends an idea of the use of personalized Markov chain factorization for next-basket recommendation, which was proposed in [14] for the recommendation of next new POIs. The authors assume locality of users' movements to their previous check-in history. The proposed matrix factorization approach incorporates the localized region constraint and the personalized Markov chain.

The next group of methods of location recommendation is based on the use of *neural networks*. In [15], the authors extend the architecture of an RNN. The authors introduce spatial and temporal elements to the network's architecture. The proposed solution addresses the deficiencies of an RNN in location recommendation: RNNs cannot model local temporal contexts well and are not capable of modelling the continuous geographical distance between locations. An RNN is applied to location recommendation in the semantics-enriched recurrent model (SERM) approach offered in [16]. The SERM distinctively enriches GPS location information with contextual text data obtained from social platforms in order to offer better prediction results. Reference [17] further improves the recommendation ideas given in [15] through the use of long short-term memory (LSTM) architecture so as to simultaneously model user's short-term and long-term interests. In reference [18], the authors modify the structures of an RNN and LSTM to incorporate different contexts (social, temporal, spatial) in the hidden and output layers to produce better recommendations. Reference [19] proposes a content-aware POI embedding model using text information about POIs (e.g., from Instagram). The proposed model consists of two context layers: check-in and text. The check-in context layer makes sure the POIs in a sequence are close enough, and the text layer is designed to capture the characteristics of POIs from the text describing them. Reference [1] focuses on the multi-level periodicity of human behavior. To incorporate this into the location recommendation model, a historical attention model is proposed and incorporated into a recurrent neural network. First, historical spatiotemporal features are extracted. They are then selected by the current mobility status to generate the most related context. By combining this context with the current mobility status, the model recommends locations based both on the sequential relation as well as on the historical regularity.

Recent approaches use attention-based models. Reference [20] uses a spatiotemporal dilated convolutional generative network for POI recommendation. The advantage of using such a methodology is the possibility of using parallel computation within a check-in sequence and thus reducing the training and evaluation times of the model. The model contains modules responsible for modeling user's geographical distance preferences by adjusting the distance in the proposed recommendations to the user's past behaviors and modeling the user's time preferences related to different categories of visited places. Reference [21] proposes an attentional recurrent neural network (ARNN) framework to improve personalized next-location recommendations in location-based social networks (LBSNs), addressing the challenge of data sparsity. A knowledge graph incorporating geographical, semantic, and user preference factors was built, and meta-path-based random walks were used to discover similar locations ("neighbors"). The ARNN integrates neighbor relationships with sequential user behavior through an attention mechanism and LSTM network to predict next locations. Real-world datasets from Foursquare (New York, Tokyo) and Gowalla (San Francisco) were used in experiments. Reference [22] re-evaluates the use of pre-trained language models (PLMs) in sequential recommendation (SR), highlighting their underutilization and redundancy in behavior sequence modeling. It proposes a simplified yet effective framework that leverages behavior-tuned PLMs for item embedding initialization, combining them with lightweight sequence models like SASRec and

BERT4Rec. Experiments on real-world datasets demonstrated that this approach improves the recommendation performance significantly without incurring additional inference costs. Reference [23] introduces MCN4Rec, a multi-level collaborative neural network for next-location recommendation. It addresses challenges like data sparsity, cold starts, and capturing complex correlations in location-based social networks. MCN4Rec integrates a multi-level view representation learning module with level-wise contrastive learning to model user–POI interactions, incorporating temporal and activity semantics. A causal encoder-decoder framework processes check-in sequences for next-location prediction. Experiments on four real-world mobility datasets showed improvements in recommendation accuracy compared to state-of-the-art baselines.

There are also approaches to recommending locations that use different techniques, other than the ones already mentioned, with interesting observations influencing the process of their use. For example, [24] considers the task of predicting the next location of a moving object. The author uses a data mining approach to predict user movement. The idea is based on association rules mining. Firstly, frequent trajectories are discovered and then transformed into movement rules. Next, the movement rules are matched to the trajectory of a moving object to determine its current location. Reference [25] analyzes human mobility patterns based on cellular carrier data and LBSNs, especially those related to friends. They find that places within a short distance that have been visited by friends may have an influence on our future location choices. They also claim that friends can influence our long-distance travels in that we are more likely to choose a distant destination because a friend of ours lives there. Reference [26] considers the problem of discovering geographical regions that are visited periodically by users of LBSNs. In particular, they aim to discover clusters of places in which a user regularly shows up, as well as the frequency related to each such cluster. The authors propose an approach to solving this task that is based on a Bayesian non-parametric model. Other research aims to first discover periodicity in the movement of objects that can subsequently be used to predict further locations. A Periodica method offered in [27] is one such example. The Periodica method first discovers periods in the movements of users as well as reference spots and subsequently attempts to find the periodic behaviors of objects within the reference spots.

The presented literature overview confirms that location recommendation is a topic that is important and interesting for researchers, yet has many sides to it. The discussed works propose approaches to location recommendation based on different paradigms. Some of them focus solely on algorithms that find the best next place to recommend, assuming that there is an existing LBSN dataset at hand. Others try to additionally consider system users' contexts (are they alone, with friends, is text information on the POIs available, etc.), resulting in more available information that can, in effect, potentially lead to better recommendation results. The presented studies focus on creating efficient methods for next-location recommendation. Our goal is to see how input data preprocessing and transformation can influence the obtained recommendation results. For this purpose, we conduct a series of experiments with modified data using the most common approaches to next-POI recommendation found in the literature. The defining characteristic of the proposed method is that it is based mainly on the preprocessing stages of data. After the data is preprocessed, one can attempt to use various models for location recommendation (specifically the GRU networks and HMMs used in this work).

3. Recurrent Neural Networks and Hidden Markov Models in Recommendation

3.1. Recurrent Neural Networks

Neural networks are among the most recognized and the most frequently used models in machine learning. They are often irreplaceable in situations in which complexity and nonlinearity play an important role [28]. They have also found their use in recommendation systems and are successfully used in commercial applications, of which a good example is YouTube, the most popular online service for video sharing [29]. In location recommendation, we deal with sequences of events. Plain, unidirectional neural networks are not capable of processing sequential information; they only process singular and independent input data. If we want to make a model sensitive to the sequential nature of data, we use recurrent neural networks (RNNs) [28]. Such networks contain feedback (a recurrent connection), allowing for signal propagation between layers backwards and over time. This means that they can store information on the history of information that has been input. By computing output value y_t in time t, they take into account not just input value x_t but also value x_{t-1}. Recurrent neural networks are thus useful in solving problems related to processing image and video data, speech recognition, language modelling, time series analysis, and location recommendation based on sequences of prior events.

In practical implementations of recurrent neural networks, neurons of special construction are used which make learning long dependencies in input data easy. The most popular gates are the long short-term memory (LSTM) and gated recurrent unit (GRU) gates. The GRU gate, due to having a lesser complexity in comparison to the LSTM gate, is slightly faster to train, and results which can be obtained with both of them are comparable and data-dependent. In our experiments we use a recurrent neural network based on GRU gates and, thus, we briefly characterize their structure.

The GRU cell [30] is a simplified version of the LSTM cell. The LSTM cell has four layers. The main layer is the one that outputs $g^{(t)}$. Its role is to analyze the current inputs and the previous state, $h^{(t-1)}$. The other three layers of LSTMs are gate controllers. As in the GRU cell, there are two gate controllers, and GRU cells have three layers that correspond to the layers of the LSTM gate. In the GRU model, there is one state vector, the so-called activation vector $h^{(t)}$, and there are two gates: the update gate, whose value is computed according to the formula for $z^{(t)}$ given in Equation (1) and which controls what part of the previous activation value will be added to the current activation state; the reset gate, whose value is computed according to the formula for $r^{(t)}$ given in Equation (1) and which controls what part of the previous information to forget.

In the GRU cell, state vectors are merged into a single vector, $h^{(t)}$. The forget gate and the input gate are controlled by a single gate controller. The full state vector is output at every time step, as there is no output gate [31].

The cell's state at each time step for a single instance is computed according to the formulas given in Equation (1).

$$\begin{aligned} z^{(t)} &= \sigma(W^{(xz)T} \cdot x^{(t)} + W^{(hz)T} \cdot h^{(t-1)}) \\ r^{(t)} &= \sigma(W^{(xr)T} \cdot x^{(t)} + W^{(hr)T} \cdot h^{(t-1)}) \\ g^{(t)} &= \tanh(W^{(xg)T} \cdot x^{(t)} + W^{(hg)T} \cdot (r^{(t)} \otimes h^{(t-1)})) \\ h^{(t)} &= (1 - z^{(t)}) \otimes h^{(t-1)} + z^{(t)} \otimes g^{(t)} \end{aligned} \tag{1}$$

In Equation (1), $z^{(t)}$ represents the update gate, $r^{(t)}$ represents the reset gate, $g^{(t)}$ represents the candidate hidden state, $h^{(t)}$ represents the short-term state, and $h^{(t-1)}$ represents the previous short-term state; $W^{(xz)}$, $W^{(xr)}$, and $W^{(xg)}$ are the weight matrices of each of the three layers for their connection to the input vector; $W^{(hz)}$, $W^{(hr)}$, and $W^{(hg)}$ are the weight

matrices of each of the three layers for their connection to the previous short-term state; $x^{(t)}$ represents current inputs.

We present a single GRU unit in Figure 1.

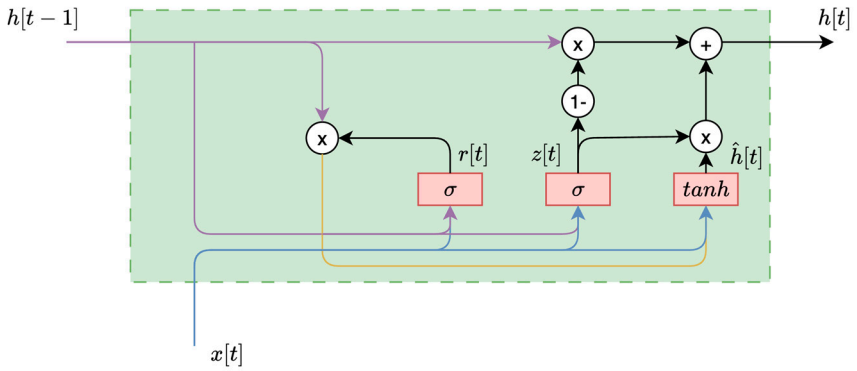

Figure 1. A single GRU unit.

As well as the gate type, there are other parameters that strongly influence the final result. The most important configurable parameters of a neural network are as follows:

- The number of hidden layers;
- The number of neurons in the hidden layers;
- The number of epochs, meaning the number of times the network will be trained with the training dataset; too large a value of this parameter can result in overfitting;
- The number of packets in each iteration, representing the number of input sequences after which weight values in the network will be updated;
- The dropout rate, preventing overfitting; the higher the value of this parameter, the higher the frequency with which random weight values in a given iteration will not be updated.

3.2. Hidden Markov Models

Hidden Markov models are based on an extended version of Markov chains. Markov chain is a model describing probability of occurrence of a sequence of given variables or states coming from some set. Examples of such sets include words, sentences, tags or symbols representing different things like the weather [32]. This model assumes strongly that in predicting a future state the knowledge of the present state is sufficient and all prior states do not influence the future. This assumption can be written in the form of an equation (Equation (2)).

$$P(X_t = x_t \mid X_{t-1} = x_{t-1}, \ldots, X_2 = x_2, X_1 = x_1) = P(X_t = x_t \mid X_{t-1} = x_{t-1}) \quad (2)$$

where X_1, X_2, …, X_t is a sequence of random variables; x_1, x_2, …, x_t are values in the state space.

Markov chain is composed of connected states. Some values of probability are assigned to both the states denoted as nodes as well as to the transitions between them. They are probabilities of occurrence of a state of a given incoming node as a consequence of a state from an outgoing node. The sum of all probability values outgoing from a node is always equal to 1. It also refers to the sum of initial probabilities. This property guarantees the correctness of Equation (2) [33]. Having transition probabilities matrix for all states, initial probability vector and taking into consideration that a given state depends only on the previous state one can compute the probability of every sequence of states using the formula in Equation (3).

$$\begin{aligned} &P(X_1 = x_1, X_2 = x_2, \ldots, X_t = x_t) = P(X_t = x_t \mid X_{t-1} = x_{t-1}) \cdot P(X_1 = x_1, X_2 = x_2, \ldots, X_{t-1} = x_{t-1}) \\ &= P(X_t = x_t \mid X_{t-1} = x_{t-1}) \cdot P(X_{t-1} = x_{t-1} \mid X_{t-2} = x_{t-2}) \cdot \ldots \cdot P(X_2 = x_2 \mid X_1 = x_1) \cdot P(X_1 = x_1) \end{aligned} \quad (3)$$

Hidden Markov models (HMM) extend Markov chains with unobservable states. It is an additional layer of states that allows for hidden states analysis, e.g., in text processing, when analyzing sentences, tags for parts of speech can be unobservable states and words can be observable states. Like in the case of Markov chains, the Markov property expressed by Equation (2) holds. An additional assumption is observation independence, stating that the probability of an observation depends solely on the state that creates it [33]. This means that the occurrence of an observation is not influenced by other states and other observations.

HMMs are characterized by the following parameters [34]: (i) N, the number of states in the model, in which the individual states are denoted as $S = \{S_1, S_2, \ldots, S_N\}$; (ii) M, the number of distinct observation symbols per state, denoted as $V = \{v_1, v_2, \ldots, v_M\}$; (iii) the state transition probability distribution $A = \{a_{ij}\}$ for $a_{ij} = P[q_{t+1} = S_j \mid q_t = S_i]$, $1 \leq i, j \leq N$; (iv) the observation symbol probability distribution in state j, $B = \{b_j(k)\}$ for $b_j(k) = P[v_k$ at $t \mid q_t = S_j]$, $1 \leq j \leq N$, $1 \leq k \leq M$; (v) the initial state distribution $\pi = \{\pi_i\}$ for $\pi_i = P[q_1 = S_i]$, $1 \leq i \leq N$. A complete specification of an HMM requires the specification of two model parameters (N and M), the specification of observation symbols, and the specification of the three probability measures A, B, and π. A compact notation is used to indicate the complete set of parameters of the model: $\lambda = (A, B, \pi)$.

There are three problems associated with hidden Markov models [34]: (i) Given a sequence of observations O and a model $\lambda = (A, B, \pi)$, we must compute the probability $P(O \mid \lambda)$ of the observation sequence. To solve this problem, the forward-backward algorithm is used. (ii) Given a sequence of observations O and the model λ, we must find the most probable state sequence Q that is optimal in some meaningful sense. To solve this problem, the Viterbi algorithm is used. (iii) Given the model parameters $\lambda = (A, B, \pi)$, we maximize $P(O \mid \lambda)$. To solve this, problem the Baum–Welch algorithm is used.

When creating a recommendation system based on hidden Markov models, one has to solve the first of the abovementioned problems. As we mentioned, the problem is solved using the forward-backward algorithm, which allows us to find $P(O \mid \lambda)$. Direct computation of this value assumes summing up the probabilities for all possible state sequences. The computational complexity in such an approach for T observations and N states would be equal to $T \cdot N^T$. This is the reason why dynamic programming, a more effective method, is typically used.

In the forward algorithm, an auxiliary variable $\alpha_t(i)$, called the forward variable, is used [33]. The forward variable is denoted as follows:

$$\alpha_t(i) = P(o_1, o_2, \ldots, o_t, q_t = i \mid \lambda) \quad (4)$$

where $o_1, o_2, \ldots, o_T$ is a partial observation sequence. The recursive step of the algorithm is expressed by the following relationship:

$$\alpha_{t+1}(j) = b_j(o_{t+1}) \sum_{i=1}^{N} \alpha_t(i) o_{ij}, 1 \leq j \leq N, 1 \leq t \leq T-1 \quad (5)$$

with $\alpha_1(j) = \pi_j b_j(o_1)$, $1 \leq j \leq N$. The recursion expressed as Equation (5) allows for calculating $\alpha_T(i)$. The required probability is thus given by Equation (6):

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \quad (6)$$

The second way to calculate $P(O \mid \lambda)$ is by using the backward algorithm. It utilizes the backward variable $\beta_t(i)$:

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, \ldots, o_t \mid q_t = i, \lambda) \quad (7)$$

Given that the current state is i, $\beta_t(i)$ is the probability of the partial observation sequence o_{t+1}, o_{t+2}, …, o_T. The recursive step of the algorithm is expressed by the following relationship:

$$\beta_t(j) = \sum_{j=1}^{N} \beta_{t+1}(j) o_{ij} b_j(o_{t+1}), 1 \leq j \leq N, 1 \leq t \leq T-1 \quad (8)$$

with $\beta_T(i) = 1$, $1 \leq i \leq N$. Consequently, we have:

$$\alpha_t(i)\ \beta_t(j) = P\{O,\ q_t = i|\lambda\}, 1 \leq j \leq N, 1 \leq t \leq T \quad (9)$$

$P(O \mid \lambda)$ can thus be computed using either a forward or backward variable:

$$P(O|\lambda) = \sum_{i=1}^{N} P\{O, q_t = i|\lambda\} = \sum_{i=1}^{N} \alpha_t(i)\beta_t(i) \quad (10)$$

4. Improving Locations Recommendation

Based on the preliminary results of location recommendations [35,36], from LBSN data, we propose ways in which such recommendations can be improved. The approaches to improving the recommendations encompass initial data preprocessing and grouping data into personalized datasets.

4.1. Improving Recommendations Quality Through Data Preprocessing

In this work, we collected and tested methods of input data preprocessing that can positively influence the generated location recommendations. Firstly, we keep only data entries that are essential from the viewpoint of further processing in location recommendation systems. We eliminate tuples that cause recommendation distortions.

We used and tested three methods of input data preprocessing, which we think can positively influence the final results:

- Division of the dataset into sequences: This is a solution used in location recommendation. A location is recommended based on a history of previously visited locations, not just based on one previous location. A basic use case, however, does not take into consideration how distant of a history of the user one has to analyze to achieve the most precise results. Because of this, we verify the influence of sequences' length on the recommendations. Sequences will be divided into sub-sequences of a specified length and tested;
- Restructuring the hierarchy of locations' categories: The authors of [35–38] observed that recommending location categories instead of concrete locations gives much better results. Modifications in the hierarchy of categories, in particular reduction of the number of categories, improves precision. This causes, however, reduction in the diversity of the final recommendations. In [35,36], the authors manually decreased the number of categories from about 800 to 20 and 17, respectively. Such a reduction results in the possibility of recommending only about 20 instead of 800 possible categories to the user. To test how the number of categories influences the recommendation results, we decided to automate the process of restructuring the hierarchy of location categories so as to have a number of categories that allows both for satisfactory recommendation precision and for a high diversity of recommendations;

- Adding an additional time-of-day field to each data tuple: Each check-in tuple of a user contains information on time, longitude, and latitude. The time information allows for specifying the time-of-day when the location was visited. Using this additional information allows for a more precise specification of the user's future preferences, but makes it more difficult to recommend a new location. For example, having 20 categories and five time-of-day intervals, we have 100 possible recommendations to present. This leads to deterioration of the final results. To overcome this problem, we added an additional time-of-day field only to the input sequence entries to generate recommendations using just information based on places' categories. Such an approach, inspired by a system based on hidden Markov models [36], produces satisfactory improvement in the model using recurrent neural networks.

4.2. Improving Recommendations Quality Through Personalized Datasets

Narrowing down the input dataset to users sharing common features allows the achievement of precise recommendations. To this end, we created personalized sets of input data following two approaches: (i) datasets based on direct information contained in the data, and (ii) individual models based on a deeper analysis of user data.

Data records cleaned after initial processing contain entries such as the user's sex or the language they use. This direct information naturally defines the user's membership in a given group. These memberships can be used to create multiple datasets, allowing for the creation of recommendations based on models trained for users who are linked by common features. We used this approach to see how it influences the generated recommendations. For this purpose, multiple test sets were created that were filtered in different ways. Next, we selected the data that were the most numerous and the most representative for each of the grouped datasets. Datasets narrowed down in such a way were used in both recommendation approaches.

It is possible to generate personalized models based on different criteria. We created models using similarity in users' locations and preferences based on their history of visited locations. Such models will be generated from the previously grouped data based on direct similarities.

To easily compare both recommendation systems, we used the same implementation environment. This allowed for easier usage of the generated test datasets and the creation of common methods that implement previously selected evaluation metrics. The use of a consistent evaluation method allowed us to point out best practices that improve the resulting recommendations and the best recommendation system.

5. Dataset Characteristics

The location recommendations considered in this paper are created from data collected from Twitter and Foursquare between 22 June and 15 July 2018. They contain users' check-ins from different parts of the world. The structure of a sample check-in is presented in Listing A1 in Appendix A.

The raw dataset of users' check-ins contains 2,483,713 entries. The data are unprocessed and, in this form, cannot be used practically.

5.1. Check-In Sequences

The first stage of data preprocessing consisted of flattening their structure and leaving only attributes useful in further processing. The new structure is observably simpler and contains only the most important information. The new structure is shown in Listing A2 in Appendix A.

We removed entries containing empty fields, in particular empty category entries. The number of data entries in the dataset after this operation was reduced to 2,476,244. We also removed data related to (i) users who were rarely active, (ii) bots, (iii) and entries that do not form time sequences.

Users that were rarely active were users that checked in less than 10 times in the considered time frame, i.e., such users who generated less than 10 data entries. Bot-generated data were data entries that were generated more often than twice within a 60 s period. In the last step, we assumed that data entries have to be located within a given data window, i.e., they have to be neighbors with at least one other entry assigned to that user. For consecutive pairs, a user-timestamp $(c_1, \tau_1), \ldots, (c_N, \tau_N)$ data entry i was removed if it did not fulfill the conditions $\tau_i - \tau_{i-1} < T$ and $\tau_{i+1} - \tau_i < T$, where T is a time window set to 8 h. These operations allowed us to eliminate data entries that were not involved in meaningful check-in sequences.

The further data preprocessing that we conducted covers additional steps: (i) data preprocessing relative to check-in sequences, (ii) data preprocessing relative to location categories, (iii) adding time-of-day information.

According to [39], the ordering of visited locations has significant importance in recommending these locations. Thus, the recommended locations are generated based on sequences of recently visited locations by a given user. To this end, the data were sorted by users and the data entries assigned to each user were grouped into sequences. Subsequent entries in sequences are located in a time window, as previously explained. For sequential recommendations to be meaningful, those with less than three entries were removed. In this way, we obtained 89,617 sequences for 28,687 users. The longest sequence consisted of 105 data entries. The average sequence length was 4.8. After performing this operation, the amount of data decreased significantly.

For place recommendations, we used sequences of consecutively visited locations where the input data are all entries, with the exception of the last one in the sequence, and the output data are the respective last entries in the sequences. We needed data containing sets of subsequences with the same constant number of records. Therefore, from the sets of sequences produced earlier, new subsequences of a given length were generated. For example, for sequences of subsequent locations [Shop & Service, Convenience Store, Park, Spiritual Center, Restaurant], one can generate three subsequences of length 3 [[Shop & Service, Convenience Store, Park], [Convenience Store, Park, Spiritual Center], [Park, Spiritual Center, Restaurant]], two subsequences of length 4: [[Shop & Service, Convenience Store, Park, Spiritual Center], [Convenience Store, Park, Spiritual Center, Restaurant]], or one subsequence of length 5: [[Shop & Service, Convenience Store, Park, Spiritual Center, Restaurant]]. Because the average length of a sequence from all the data is 4.8, we used three different datasets with subsequences containing three, four, and five entries, and verified the influence of their length on the results.

5.2. Analysing Locations Categories and Adding Time-of-Day Attribute

The recommendation systems discussed in this paper do not generate propositions of specific places but rather place categories, which, according to [40], offers considerably better recommendation results. It is thus vital to prepare the categories correctly. Foursquare location data are assigned to about 930 categories that are stored in a tree-like structure containing at most five levels. Based on experiments done in [35,36], the adjusted categories were either too detailed or too general. What is more, the categories created in these works were prepared created manually. According to [41], higher-level categories output better results, as users typically share only 10% of their location data, which results in difficulty in finding similarities in preferences among individual users when using low-level categories.

Based on the above-mentioned observations, we automated the restructuring of category fields in the dataset. All records were, first, sorted according to their *category_id*. If the number of records related to a given category was smaller than a given threshold, that category was replaced with the category of the higher level. Records not fulfilling the threshold assumption were completely deleted from the dataset. The threshold was computed as the number of records for the n-th most numerous category in the dataset, where n can be modified for test purposes.

Using this approach, we generated three datasets to test the influence of the number of categories on the results. The first dataset contains all 834 different categories, while the second and the third one have 45 and 20 categories, respectively.

Table 1 presents the most numerous and the least numerous categories after restructuring in the case of limiting the number of categories to 34. Most of the generated categories consist of subcategories. For example, "Japanese Restaurant" includes records drawn from as many as 18 different subcategories, such as "Tonkatsu Restaurant", "Sushi Restaurant", and so on.

Table 1. Number of records for the most and the least numerous categories after category restructuring.

Category	Contains Subcategories	No of Records
Most numerous categories		
Train Station	No	85,424
Shop & Service	Yes	58,938
Food	Yes	56,372
Least numerous categories		
Park	No	6065
Residential Building (Apartment/Condo)	No	5830
Spiritual Center	Yes	5728

Some of the resultant categories can be uninteresting from the location recommendation perspective, e.g., "Train Station" or "Home". At later stages of creating recommendation systems, such records were removed from the dataset. This can, however, lead to distortion and deterioration of the results. In particular, the category "Train Station", being very numerous, can have a significant impact on the final results. Additional tests were thus be conducted to verify the influence of this category on the recommendation quality.

Based on conclusions from [37] suggesting a significant improvement in recommendations when including the time of day information for check-ins, we added a new attribute with information on the time of day when a user visited a given place. The specific time of the day was generated based on the timestamp, from which hour in the day was extracted using the intervals given in Table 2. The time zone was also taken into account, based on the longitude and latitude information of a given check-in.

Table 2. Time of the day definitions used in the additional attribute.

Time of the Day	Hourly Intervals
Morning	5.00–9.00
Noon	9.00–14.00
Afternoon	14.00–19.00
Evening	19.00–24.00
Night	24.00–5.00

5.3. Preparation of Datasets

In [35–37,42], the authors determined that recommendation systems using data personalized for every user enable the achievement of much better results. Using user sex and language information, we generated datasets characterized by these attributes. In this, way we generated two datasets of female and male users. As far as user languages are concerned, the dataset contained 38 languages, out of which we focused on the two most numerous: Japanese and English. Additionally, we created datasets that narrowed users down according to both sex and language. The data cardinality for each dataset is given in Figure 2. For each data division criterion, the most numerous dataset was selected for use in the tests. The selection encompasses the following: (i) a general dataset, (ii) a dataset of male users, (iii) a dataset of users speaking Japanese, and (iv) a dataset of male users speaking Japanese.

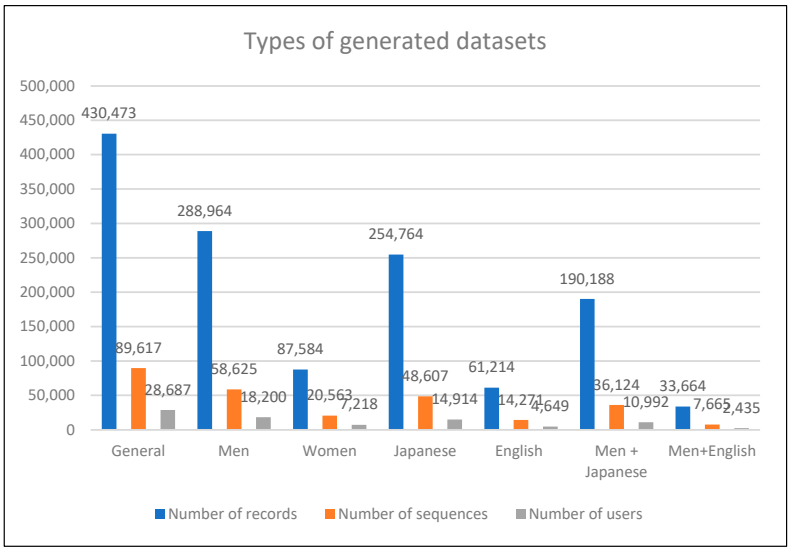

Figure 2. Number of records, sequences, and users for different generated datasets.

5.4. Individual Datasets

One of the observations made by the authors of [38] was that using models generated particularly for a user for whom a recommendation is created had a very positive influence on the recommendation. To that end, we used individual models.

Implicit profiles for each user were generated by counting the number of occurrences of each location category where the user checked in. Vectors created in this way have a length equal to the maximal number of unique, possible categories. The vectors store information on how often a given user stayed in a location with an assigned category. The larger the value of a given element of the vector, the more the user preferred locations of that category. The sequence of categories in each vector was constant and identical for all users. The vectors were then put in a matrix, where each row represented a user's profile. Next, similarities among these vectors were computed. For this purpose, we used a cosine similarity measure. The threshold value for the similarity of users was set to 0.7. The personalized model for each user was determined by computing the cosine similarity of each user in relation to those of all other users and selecting only those exceeding the predetermined threshold.

We also generated individual models based on users' location. The model for a given user was determined as all other users located in the longitude and latitude range of $< -1, 1>$ in relation to that user. This translates to other users who are at a distance of roughly 111 km from the given user.

5.5. Training and Testing Datasets

In the case of regular datasets, the division into training and testing parts was done randomly in the proportion of 7:3. For the individual datasets for a given user, we used sequences of all users with similar profiles. A test set consists of sequences assigned to the analyzed user.

5.6. Data Preprocessing: A Summary

One of the main conclusions from the literature analysis was that correct data preprocessing has a significant influence on the outcomes of location recommendation. To that end, we utilized the experiences of the authors of [35,36] in this regard and introduced some improvements.

The generated general dataset for sequences of the minimal length of three and divided into 34 categories containing 430,473 records grouped into 89,617 sequences for 28,687 users. The number of data entries in the respective stages of data preprocessing are given in Figure 3.

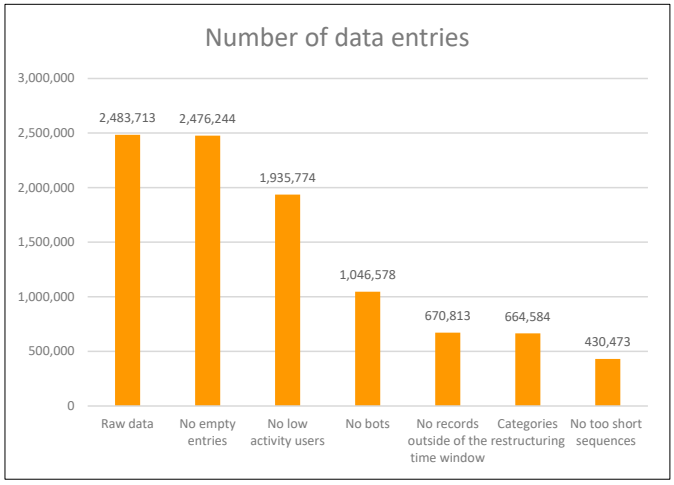

Figure 3. Number of data entries after each preprocessing step.

The conducted data preprocessing steps are summarized in Figure 4.

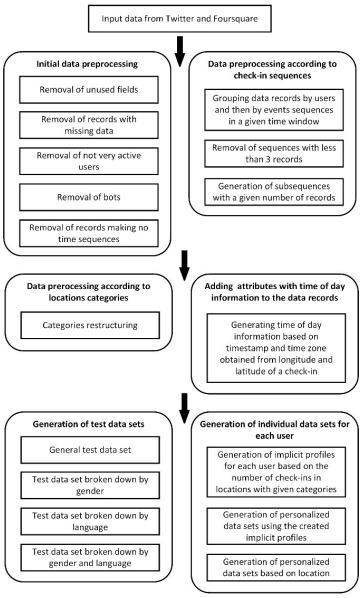

Figure 4. Subsequent stages of data processing.

6. Computational Cost and Scalability

Knowing the details of the proposed preprocessing stages, we would like to assess their computational cost, including also the two methods used for location prediction: GRUs and HMMs.

First, let us consider the cost of the preprocessing stages presented in Figure 4. *The initial data preprocessing* stage can be performed in liner time frame. This initial stage consists of steps such as: the removal of unused fields, the removal of records with missing data, and the removal of activity generated by bots. Generally, such steps can be performed by scanning the input data only once.

Subsequently, the *data are preprocessed according to check-in sequences*. First, one needs to apply a sorting algorithm to group the entries according to their user ID number. Subsequently, sorting is applied to the event sequences of each user. For sorting, the Quicksort algorithm, for example, can be applied, which typically achieves an O(N log N) time complexity, where N is the size of the dataset to be sorted. The final step of sequence preprocessing consists of the generation of subsequences with a given number of records. This can be achieved by scanning the event sequence of each user once.

In the next stage, *data preprocessing according to location categories*, the number of recommendation categories is reduced in the dataset. To this end, the automatic process is applied that replaces categories with a cardinality below a given threshold with their super-categories. Knowing how numerous each category is, this step can be achieved in a linear time frame. Similarly, the stage of *adding the time-of-day attribute* to each record can be achieved by scanning the dataset only once.

For *generating the training/testing data*, non-individual datasets can be easily created by selecting data based on specific attributes, such as gender or language, which can typically be done in a constant or linear time frame relative to the dataset size. The *generation of training and testing datasets for individual (personalized)* profiles is more computationally demanding. For the implicit profiles, first, a matrix containing the categories of locations visited by each user needs to be calculated. This can be achieved, again, by scanning the input dataset once. Subsequently, the cosine similarity is calculated for each pair of rows in the matrix, which generally requires a quadratic number of operations relative to the size of the matrix. To obtain location-based personalized datasets, one needs to apply calculations that provide geographical distance between various visited locations of pairs of users. This can be achieved in polynomial time relative to the number of users and visited locations.

In the case of both GRUs and HMMs, the computational complexity depends on the length of the generated subsequences as well as other factors. Specifically, HMM's hidden states are generated by combining all categories of locations with the times of day obtained in the preprocessing step. As described in the Experimental Evaluation section, this can have a significant impact on the efficiency of the proposed model when HMMs are used, practically limiting the number of categories used in experiments to up to 40.

The scalability of the proposed solution is limited due to the need to train a location recommendation model (being either a GRU network or a HMM). However, at least some of the preprocessing steps proposed in this work can be implemented directly on a mobile device that is used to collect location/review data. For example, the preprocessing steps such as the initial removal of empty fields, adding the time of the day, or generating locations' subsequences can be performed on the user side rather than on a computing server. Furthermore, nowadays, training a GRU neural network or conducting matrix computations (e.g., similarity between users in a location matrix) can be significantly sped up with the use of graphical cards.

7. Experimental Evaluation

In this work, we compare two location recommendation systems using a common dataset and the same quality measures. In the following subsections, we discuss the implementation of these systems and provide the recommendation results generated by each of them. Finally, we compare the two systems.

7.1. Quality Measures

Prediction accuracy is the most commonly used quality metric. Recommendation systems themselves largely operate as predictive systems. The main premise of this metric is that a model is better if it can more accurately predict a user's decision. This metric, unlike many others, can also be successfully applied in offline tests.

The recommendation systems created in this work are based on data collected in the past from a LBSN site. This means that only offline tests of the models will be performed. This fact alone limits the quality measures that can be used. In addition, recommendations will be issued based on the categories of locations that a given user visited in the past. In this case, prediction accuracy metrics that take into account user's decision will be used.

When evaluating the tested models, we use the precision. The precision is defined as the ratio of the number of recommended categories that correspond to those that the user actually visited at a given moment in the past to the number of all recommendations [43]. Additionally, in order to extend the scope of recommended objects, we included the precision, taking into account situations where the expected category was among one of the three most likely model proposals (Precision@N [43], or *Precision@3* and *Precision top 3* in our case).

7.2. Recurrent Neural Networks Setup

The location recommendation system constructed herein using recurrent neural networks follows a similar approach to [37], and extends the model from [35]. It was created using the Keras library, which allows for easy and quick design and testing of neural networks.

To create a model, we needed to first create a three-dimensional tensor, where the respective dimensions are built up by (i) samples—subsequent data sequence, (ii) timestamps—singular sample containing events in a given time interval, and (iii) features—data characterizing a single event.

The created recurrent neural network consists of three layers, each of which is built up of 40 GRU neurons. The input data tensor consists of consecutive sequences of visited locations. Sequentially visited location categories in a given sample are time stamps, and a single category is a feature. Features were encoded using the one-hot method, where a feature is represented as a vector of length equal to the number of categories, containing zeros except for when there is a represented category, indicated by one. The output layer contains a number of neurons corresponding to the number of categories, in accordance with the one-hot encoding method, as the output value is a single predicted category. The activation function, *softmax*, normalizes the values for each output neuron so that they are in the range <0, 1> and corresponds to the probability of visiting a place of a given category. *Adam* is an advised optimizer that returns good results for a wide range of problems. The loss function is set to *categorical crossentropy*, which is used in problems related to classification with multiple categories. The parameter values of the recurrent neural network are given in Table 3. The tested aspects of the implemented location recommendation systems, along with possible values, are gathered in Table 4.

As suggested by the conclusions from [35] and previously conducted experiments with the recurrent neural network, we used constant values of network parameters for further testing. It turns out they have a limited impact on the achieved results, so we set

these parameters to values that return, possibly, the best results, taking into consideration the short network training time.

Table 3. Parameter values of the recurrent neural network.

Parameter	Used Value(s)
Neuron type for hidden layers	GRU
Number of layers and number of neurons in these layers	40-40-40
Loss value for each layer	0–0.2–0.2
Batch size	3 (2 for individual models)
Number of epochs	50

Table 4. Tested aspects of the implemented location recommendation systems along with possible values.

Tested Aspect	Possible Values
Dataset type (grouping by user type)	all/males/speakers of Japanese/males speaking Japanese
Number of categories (after hierarchy restructuring)	all/<50/<25
Sequence length	3/4/5
Including the most numerous category (Train Station)	included/excluded
Using individual models	based on similarities among users/based on distance among users
Including time of day information	included/excluded

7.3. Recurrent Neural Networks: Results

In Figure 5, we show the results for different datasets for the same network parameters. As expected, the global model returns the worst results. Additionally, it turns out that filtering using users' gender information returns a barely visible improvement. On the other hand, better results were obtained for sets of users filtered out by the language they use. Based on the obtained results, it can be observed that, when choosing places to visit, cultural aspects (which can be inferred from the used language) are much more important than the gender of the users. For further testing we will be using the dataset for which the best results were achieved, i.e., the *Japanese language* dataset.

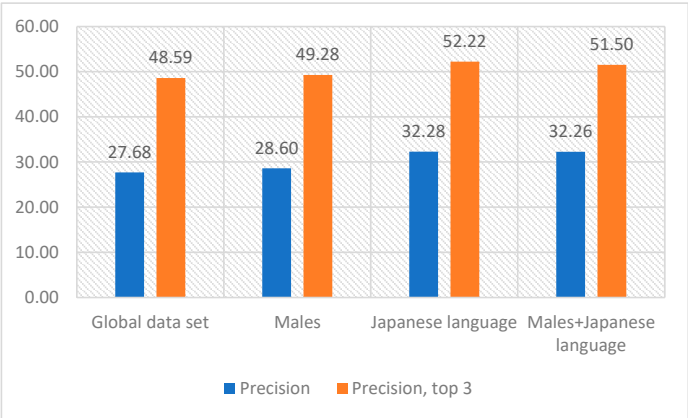

Figure 5. Precision and precision@3 for recommendations computed using RNN for different datasets. Parameters: sequence length: three, number of categories: 25, the most frequent category: included, time of day: not included.

Tests of the impact of the number of place categories on the results confirmed the results reported in [35]. Overall, we observed that the fewer categories the analyzed dataset contains, the better the recommendation results are that can be obtained. However, reducing this number leads to a deterioration in the differentiation of recommendations. Therefore, we decided to focus on 25 categories, which still allows for a much better differentiation than in [35,36]. A comparison of the results for different numbers of categories is presented in Figure 6.

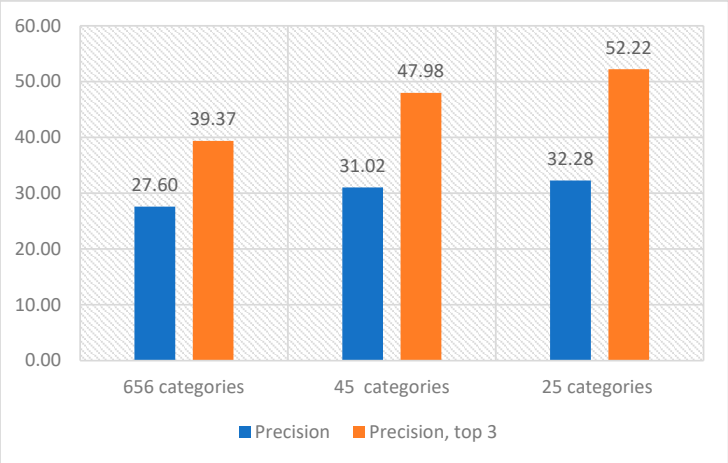

Figure 6. Precision and precision@3 for recommendations computed using RNN for different numbers of categories for the following parameters: language—Japanese, sequence length—three, the most numerous category—included, time of day information—included.

The results obtained by comparing the datasets with different sequence lengths show that the longer the sequence, the better the precision, as shown in Figure 7. However, this tendency may only apply to short sequences of events. With more places visited over time, those from the distant past may lose their relevance completely to a recommendation for a given moment. However, this cannot be verified, because in the data from LSBN services, as pointed out in [41], a very small percentage of users reports their check-ins regularly, which results in a lot of time gaps and the obtained sequences typically being short. For sequences with a length of four, the number of records obtained for analysis drops by more than 35% compared to the number of records for sequences with a length of three. For most tests, we used sequences of visited places with the minimum length of three.

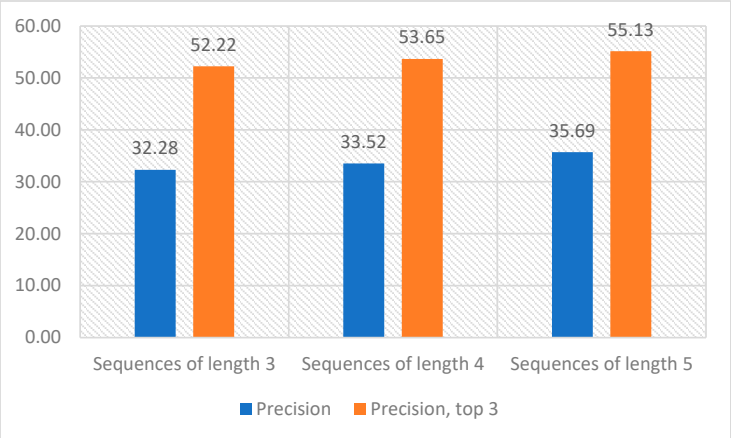

Figure 7. Precision and precision@3 for recommendations computed using RNN for different sequence lengths for the following parameters: language—Japanese, the most numerous category—included, time of day information—not included.

We also verified what values of precision can be obtained when the records with the most numerous category ("Train Station") are deleted completely or when they are deleted only when they appear as the last element in the sequence. Figure 8 shows that both of these attempts produced rather weak results.

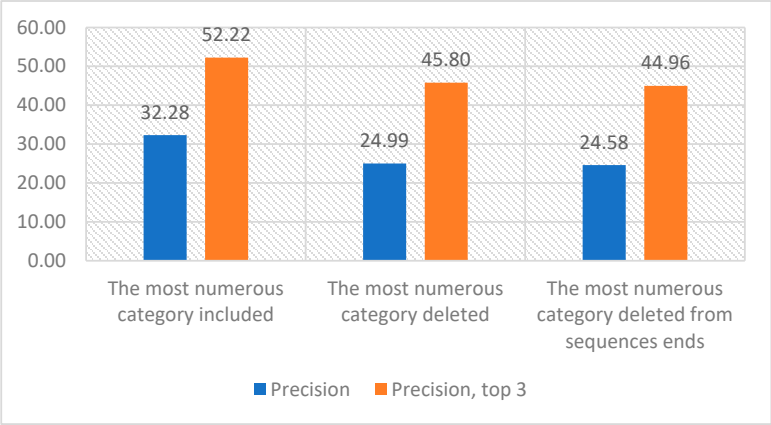

Figure 8. Precision and precision@3 for recommendations computed using RNN with and without the most numerous category for the following parameters: language—Japanese, number of categories: 25, sequence length: three, time of day information—not included.

All the models obtained for standard datasets still give unsatisfactory results. The solution that most significantly improved the results turned out to be the use of personalized models. Individual models based on the similarity of preferences between users allowed us to improve the precision by as much as 14% in relation to the global model using the same dataset. However, these tests were performed on specially selected users who had a lot of check-ins and a sufficient group of similar users (training set). Despite the high average precision, the results varied considerably from one user to the next, as shown in Figure 9. The improvement in the results was also achieved thanks to the location-based individual models. In this case, the increase in precision was 10%, which is a good result, but still weaker than in the case of models based on the similarity of preferences between users.

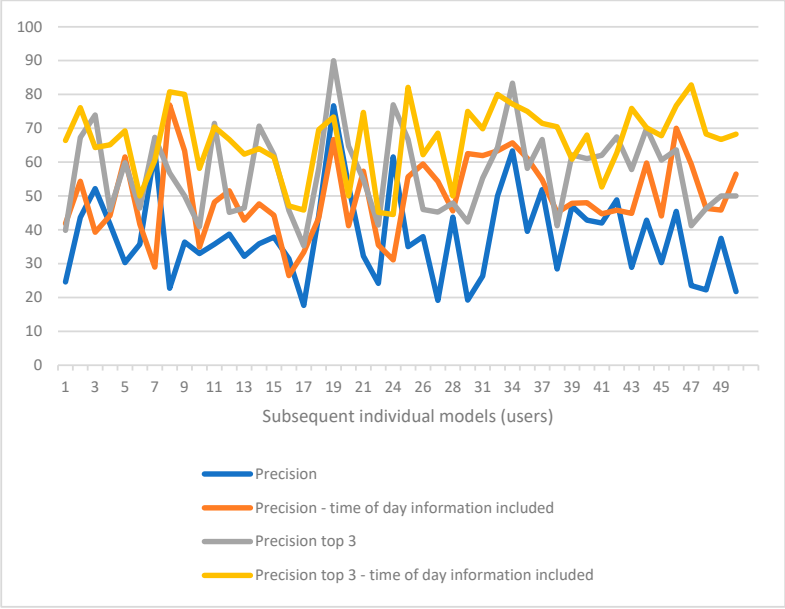

Figure 9. Results for particular individual models (similar preferences) of a recurrent neural network.

Figure 10 shows the results obtained when the tested model also took into account the time of day of each check-in. There was a slight, 1.8% improvement in the precision for the global model and a 1.3% improvement for the individual model based on the similarity of locations. However, this method allowed us to achieve very good results in the case of the personalized model in terms of preferences. The obtained precision value was 53% and the precision value for the three best categories was 72%. These are the best results so far. The improvement in the results of individual models personalized for subsequent users, taking into account the time of day, can be seen in Figure 9.

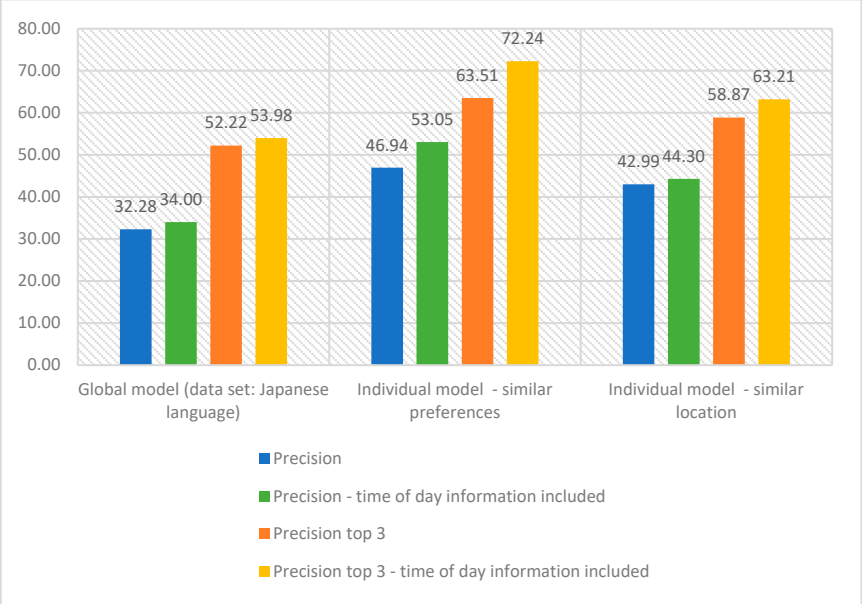

Figure 10. Influence of individual models of a recurrent neural network on precision for the following parameters: dataset for the Japanese language, time of day information—included, number of categories—25, sequence length—three, the most numerous category—included.

We made an attempt to improve the best-performing individual model based on the similarities of preferences, taking into account the time of day, by increasing the length of the sequence. Processing the data so that they had a minimum sequence length of five resulted, as already mentioned above, in a significant decrease in the amount of data to be analyzed. The individual models generated from this type of data were therefore much less numerous than before, and the results obtained were based on a smaller number of tests. However, the obtained precision value, which was 55.5%, and the precision value for the three best categories, amounting to 74.2%, confirmed our earlier assumptions and turned out to be the best configuration in the case of recursive neural networks. The results for this model are shown and compared in Figure 14.

7.4. Hidden Markov Models Setup

The location recommendation system based on hidden Markov models (HMMs) was created using the Python *hmmlearn* library. The use of the same environment as in the case of the model based on recursive neural networks allows for easier sharing of the test input sets and easier comparison of the obtained results.

In the created model, the hidden states are the categories of visited places together with the time of day in which the event took place. The observations are categories that were logged by the user last in a given sequence. Such a structure of the model allows for the use of the check-in sequence together with the time-of-day information. A fragment of the structure of the discussed hidden Markov model is presented in Figure 11.

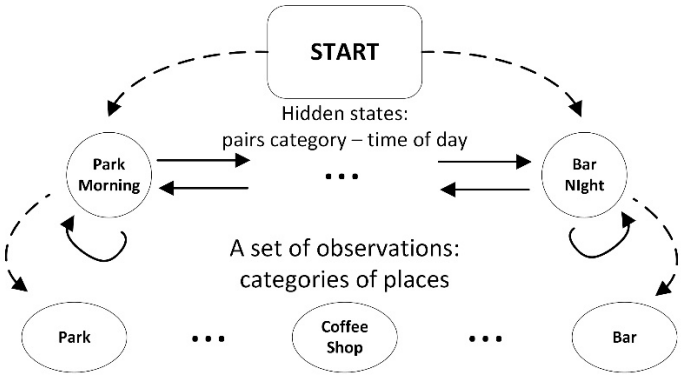

Figure 11. A fragment of the structure of the implemented hidden Markov model.

In order to compute the next visited place, the problem of evaluation should be solved. For this, initial probability, transition, and emission matrices are required. These matrices were computed as follows.

- Initial probabilities matrix: separate sets of records for each type of hidden state (attributes: category, time of day) were assembled and the size of each of those sets was divided by the total number of all records;
- Transition matrix: separate sets of records for each type of hidden state (attributes: category, time of day) were created with a corresponding hidden state preceding them, and the size of each of these sets was divided by the total number of records in the sets created from the corresponding previous attribute, category and time of day.
- Emission matrix: separate sets of records for each type of observation (category) were created with a corresponding hidden state type preceding them (attribute: category and time of day), and the size of each of these sets was divided by the total number of records in the sets created from the corresponding previous attribute, category and time of day.

The evaluation problem is solved using the *forward-backward* algorithm. This algorithm is available in the *hmmlearn* library and can be invoked with a method that returns the posterior probability matrix for each hidden state. From this matrix, probabilities assigned only to the categories corresponding to the time of day for the searched recommendation are selected.

The tests performed using hidden Markov models reflect the testing procedure executed with RNNs, so that comparison of the results is possible. Taking into account the conclusions obtained when checking the impact of the number of categories, we found that the results for the models taking into account all possible categories are unsatisfactory. In addition, with around 800 different categories and five times of day, more than 4,000 hidden states can be obtained. This causes a very high complexity of such a model, so we limited the tests to 40 and 25 different categories.

7.5. Hidden Markov Models Results

The first test performed for the recommendation system based on hidden Markov models was to determine the influence of the dataset type on the obtained results. The comparison of the obtained results is shown in Figure 12. As in the case of the recursive neural network, the precision values for the global sets and for the male users are practically identical and a few percent worse than for the set of Japanese speakers. One can also notice here a slight deterioration of the model with the combination of the sets "Men" and "Japanese language" compared to the model using only the set "Japanese language". Based on the above conclusions, our further testing mainly used the dataset for Japanese-speaking users.

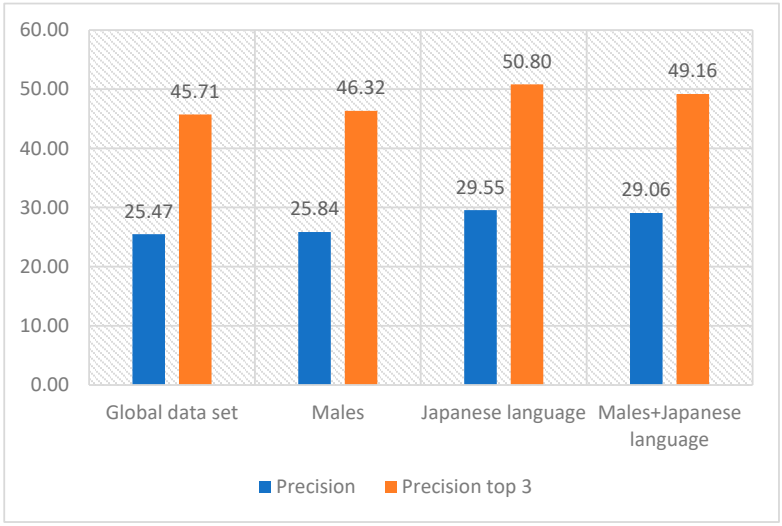

Figure 12. Comparison of precision and precision@3 for hidden Markov models for different types of datasets. Parameters: sequence length—three, number of categories—25.

Similarly to the model based on recursive neural networks, the reduction in the number of location categories resulted in a slight increase in precision. This is shown in Figure 13. In the same plot, one can also see the influence of the sequence length on the results. Here, too, the increase in the number of elements of the sequence of events from three to four resulted in slightly better results. However, for sequences with a length of five, the precision value decreased slightly. The differences between the results for the individual models with different sequence lengths are less than a percent and, unlike for the neural network models, are not that significant.

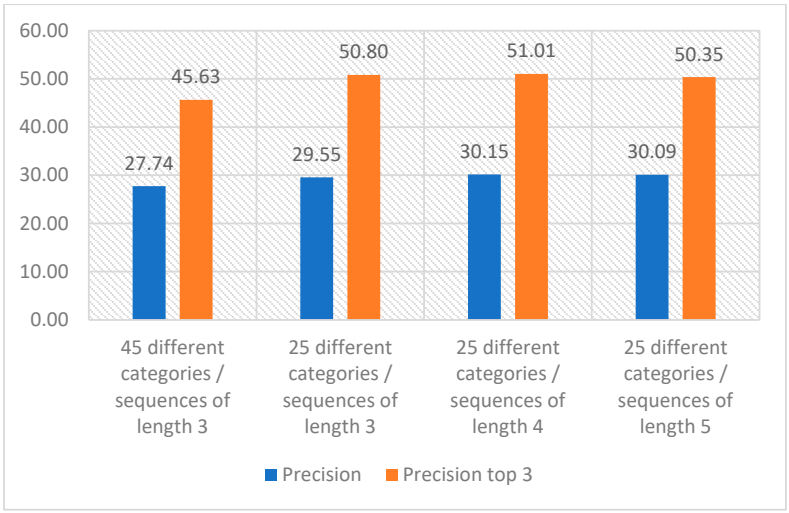

Figure 13. Comparison of precision and precision@3 for hidden Markov models for different numbers of categories and different sequence lengths. Parameters: dataset—Japanese language.

In the next stage we tested individual models. The results of these studies are presented in Figure 14. A very large increase in precision was obtained here, both for the models based on the similarity of preferences and those using location similarity. In the first case, the precision value was over 51% and the precision value for the three best categories was over 70%. In both cases, this is an increase of approximately 20% against the global model that uses the same dataset. In the case of the models based on the distance between users, this increase was slightly smaller, but also allowed us to obtain satisfactory results.

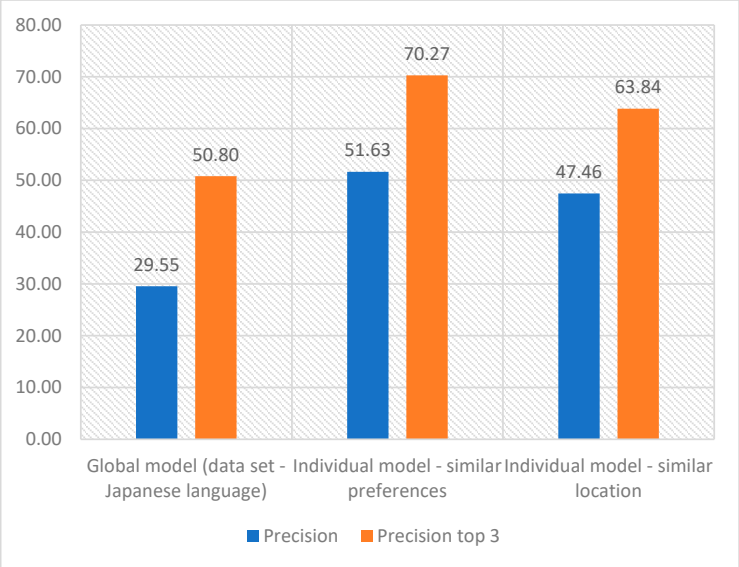

Figure 14. Influence of using individual models on precision and precision@3 with hidden Markov models. Parameters: dataset—Japanese language, categories number—25, sequence length—three.

The differences in the results between global and individual models in the case of hidden Markov models are much more visible than those for recurrent neural networks. Moreover, in both of these cases, one can observe the advantage of preferences similarity over locations similarity. It should also be noted that all tests for hidden Markov models were performed taking into account the time of day of the check-in.

To further improve the results, we generated an individual model for data with sequences of a minimum length of four and that was based on the similarity of preferences between users. The obtained values of precision and precision@3 are presented in Figure 15 and amount to 51.8% and 70.8%, respectively. These results are almost the same as in the case of data with sequences of a length of three. This means that, in the case of the recommendation system based on hidden Markov models, the effect of the sequence length is of very little importance, while the creation of personalized models allows us to achieve satisfactory recommendations.

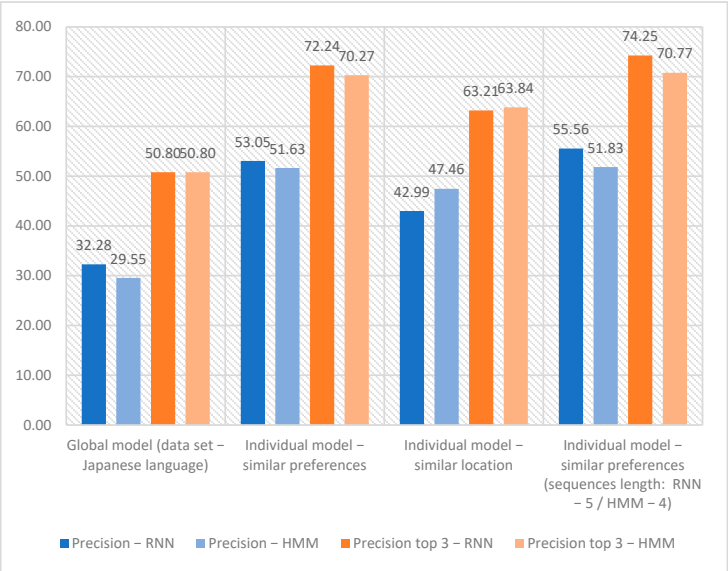

Figure 15. Precision and precision@3 comparison for models based on recurrent neural networks and hidden Markov models. Parameters: dataset—Japanese language, number of categories—25, sequence length—three.

7.6. Comparison of Recurrent Neural Networks and Hidden Markov Models

Figure 15 summarizes the results of location recommendation systems based on recursive neural networks and recommendation systems using hidden Markov models. For almost all types of models, the results were better by several percent in the case of the neural networks approach. The exception is the individual model based on location similarity. In both cases, only the creation and application of models personalized in terms of user preferences allowed us to exceed the threshold of 50% for the precision value and to exceed the threshold of 70% for the precision for the three best categories. Recommendation systems using global datasets, regardless of how they would be processed, do not give satisfactory results and it is not possible to use them in practice.

A slight improvement in the results, even for individual models, can be achieved by manipulating the length of the check-ins sequence. For the RNN models, the use of a sequence with a length of five allowed us to improve the precision by a few percent. Longer sequences are impossible to test on the collected dataset for the reasons mentioned previously. In the case of HMM models, of the tested sequence lengths, the length of four turned out to be the best.

For both types of location recommendation systems, the supposition was confirmed that better results can be obtained using a smaller number of location categories. However, these differences are not as significant as in the case of using personalized models. Therefore, in order to achieve a greater diversification of recommendations at the cost of a slight decrease in precision, it is possible to use lower-order categories and restructure the hierarchy of location categories differently.

The comparison of both systems showed that both the initial data processing and the type of recommendation system used are much less important than the use of an individual model. Manipulating data processing allows us to improve the precision by several percent. The same is the case with changing the type of recommendation system. Only the use of individual models resulted in an increase in precision by over a dozen percent, and, in the case of taking into account the time of day for RNN systems, even 20%.

8. Conclusions and Outlook

In this paper we were looking for solutions which would allow us to improve the quality of location recommendation systems based on data collected from LBSN sites. As part of the work, two location recommendation systems were created. The first one used recursive neural networks, and the second one used hidden Markov models. The structure of both of these models was based on the conclusions drawn from the works [35,36] and other scientific articles dealing with the topic of recommendations based on data from social network sites.

Among the applied solutions aimed at improving the precision of the recommendations of the created systems, the most effective turned out to be the use of personalized models based on user preferences. This solution in combination with the use of recommendations based on categories rather than specific places, taking into account the time of day of the check-in and the appropriate data processing, including the restructuring of the hierarchy of location categories, the selection of an appropriate dataset for the appropriate group of users, and the use of check-in sequences of an appropriate length, allowed us to obtain satisfactory results for the both tested systems.

The comparison of the two types of recommendation systems showed that slightly better results can be achieved with the use of recursive neural networks. However, the difference is not that significant. Much more important than the type of system used is the personalization of the model in relation to the user data.

The recommendation systems that were created, despite the fact that they allowed us to obtain precision values that were better than before, are still not good enough for practical applications. Only the use of the top three categories as a recommendation, where the highest precision value for the RNN model was 74% and that for the HMM model was 70%, could enable the use of these systems by real users. However, to achieve these precision values, it would be necessary to change the range of analyzed categories, as the solutions used in this work did not exclude categories that, from a practical point of view, do not make much sense as recommendations when determining suggested locations. These include categories such as "Train Station" or "Home," which are among the most numerous, and their omission in the recommendations significantly reduces their precision.

The conclusions from the conducted research indicate that the greatest improvement in results can be achieved through the personalization of models for individual users. This study utilized only two types of such models. Therefore, one possible direction for further development is creating other types of individualized datasets for specific users. These models could then be combined in appropriate ways to achieve the best possible results.

The further refinement of personalized models could focus on distinguishing user behavior between workdays and weekends. On workdays, activity patterns often include distinct morning and afternoon peaks, corresponding to commuting and lunchtime, with the highest activity typically being observed in the evening. In contrast, weekends exhibit different dynamics, such as the absence of a morning peak, higher activity during midday, and increased late-night engagement. Special consideration could also be given to hybrid days like Fridays, where user behavior blends characteristics of both working and non-working days. By leveraging such nuanced temporal patterns, future models could better anticipate user needs, leading to more contextually relevant and accurate recommendations.

Another development opportunity lies in the observation that, for neural networks, longer check-in sequences over time lead to improved precision. In the dataset used in this study, the average sequence length was only 4.8. It might therefore be worthwhile to collect datasets where these sequences are significantly longer for individual users and then conduct tests on them.

Additionally, by analyzing the results from studies [35,36], which were obtained using the manual restructuring of location categories, and comparing them to the results from the automated method applied in this work, it is evident that appropriately manipulating the hierarchy of categories can slightly improve the model. Further development would thus involve attempting to find a structure of location categories that allows for achieving the highest possible precision.

Author Contributions: Conceptualization, R.B.; methodology, M.O.; software, M.O.; validation, R.B., M.O. and P.M.; formal analysis, P.M.; investigation, R.B. and M.O.; resources, R.B. and M.O.; data curation, R.B.; writing—original draft preparation, R.B. and P.M.; writing—review and editing, R.B. and P.M.; visualization, M.O.; supervision, R.B.; project administration, R.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

This section provides a list of key abbreviations used throughout the paper for clarity and ease of reference.

- **ARNN** Attentional recurrent neural network—a framework for improving personalized next-location recommendations in location-based social networks.
- **BERT4Rec** Bidirectional encoder representations from transformers for recommendation —a sequential recommendation model based on bidirectional transformers.
- **GPS** Global positioning system—a satellite-based navigation system that provides location and time information.
- **GRU** Gated recurrent unit—a simplified version of RNN architecture, optimized for efficiency in learning long-term dependencies.
- **HMM** Hidden Markov model—a statistical model used to predict sequences based on observable and hidden states.
- **LBSN** Location-based social network—platforms that integrate social networking with location-based services, enabling users to share their location and discover new places.
- **LSTM** Long short-term memory—a type of RNN architecture designed to overcome the vanishing gradient problem, allowing it to learn long-term dependencies in sequential data.
- **MCN4Rec** Multi-level collaborative neural network for next location recommendation
- **ML** Machine learning—a field of artificial intelligence focused on algorithms that learn patterns from data.
- **PLM** Pre-trained language model—models trained on large text corpora and fine-tuned for specific tasks.
- **PLMs** Pre-trained language models—models that are pre-trained on large text corpora and fine-tuned for specific tasks, widely used in natural language processing and sequential recommendation tasks.
- **POI** Point of interest—a specific location that users might find interesting or relevant, such as restaurants, parks, or landmarks.
- **RNN** Recurrent neural network—a type of artificial neural network designed to handle sequential data by incorporating temporal dependencies.
- **SASRec** Self-attentive sequential recommendation—a recommendation model leveraging self-attention mechanisms.
- **SERM** **Semantics-enriched recurrent model**—SERMs jointly learn the embeddings of multiple factors (user, location, time, keyword) and the transition parameters of RNNs in a unified framework.
- **SR** Sequential recommendation—a recommendation approach that predicts the next item or action based on a sequence of past interactions.

Appendix A

The Appendix contains the structure of a sample check-in collected from Twitter and Foursquare (Listing A1). After initial preprocessing the structure of the dataset was flattened: only attributes useful in further processing were kept. The new structure is shown in Listing A2.

Listing A1. Structure of a sample data entry.

```
{
"status": {
    "createdAt": "Fri Jun 22 04:38:48 +0000 2018",
```

```
        "id": 1010019246229749760,
"text": "I'm at The Halal Guys in West Hollywood, CA
https://t.co/Zd2NlZQPTA",
"entities": {
 "urls": [
                {
                    "url": "https://t.co/Zd2NlZQPTA",
                    "expandedUrl": "https://www.swarmapp.com/c/3oayqUOTx8i",
                    "displayUrl": "swarmapp.com/c/3oayqUOTx8i"
                }
            ]
        },
        "user": {
            "id": 39206945,
            "name": "Sean Franklin",
            "location": "ÜT: 32.810707,-96.795155"
              },
        "geo": {
            "coordinates": [
            34.08464354,
            −118.38469893
            ]
        },
        "place": {
            "id": "1927193c57f35d51",
            "name": "West Hollywood",
            "url": "https://api.twitter.com/1.1/geo/id/1927193c57f35d51.json"
        },
        "lang": "en"
 },
 "checkin": {
 "id": "5b2c7d515a2c91002c3dddfe",
 "createdAt": 1529642321,
 "venue": {
        "id": "59953c0d8c35dc5eac0edc7a",39

        "name": "The Halal Guys",
        "location": {
            "address": "8919 Santa Monica Blvd",
            "lat": 34.08461040280683,
            "lng": -118.3846760374131
        },
        "categories": [
        {
            "id": "4bf58dd8d48988d16e941735",
            "name": "Fast Food Restaurant",
            "pluralName": "Fast Food Restaurants",
            "shortName": "Fast Food",
            "primary": true
        }
        ],
        "stats": {
```

```
                    "checkinsCount": 384,
                    "usersCount": 240,
                    "tipCount": 0,
                    "visitsCount": 0
                },
                "price": null,
                "likes": null,
                "rating": null,
                "ratingSignals": null,
                "primaryCategory": {
                    "present": true
                }
        },
        "user": {
                "id": "2156575",
                "firstName": "Sean",
                "gender": "male"
        }
        }
}
```


Listing A2. Flattened structure of a sample data entry.


```
{
    "category_id": "4bf58dd8d48988d16e941735",
    "category_name": "Fast Food Restaurant",
    "gender": "male",
    "id": "5b2c7d515a2c91002c3dddfe",
    "language": "en",
    "latitude": 34.08461040280683,
    "longitude": -118.3846760374131,
    "timestamp": 1529642321,
    "user_id": "2156575",
    "venue_id": "59953c0d8c35dc5eac0edc7a"
}
```


References

1. Feng, J.; Li, Y.; Zhang, C.; Sun, F.; Meng, F.; Guo, A.; Jin, D. DeepMove: Predicting Human Mobility with Attentional Recurrent Networks. In Proceedings of the 2018 World Wide Web Conference on World Wide Web—WWW '18, Lyon, France, 23–27 April 2018; ACM Press: New York, NY, USA, 2018; pp. 1459–1468. [CrossRef]
2. Gan, M.; Gao, L. Discovering Memory-Based Preferences for POI Recommendation in Location-Based Social Networks. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 279. [CrossRef]
3. Averell, L.; Heathcote, A. The form of the forgetting curve and the fate of memories. *J. Math. Psychol.* **2011**, *55*, 25–35. [CrossRef]
4. Loftus, G.R. Evaluating forgetting curves. *J. Exp. Psychol. Learn. Mem. Cogn.* **1985**, *11*, 397–406. [CrossRef]
5. Murre, J.M.J.; Dros, J. Replication and Analysis of Ebbinghaus' Forgetting Curve. *PLoS ONE* **2015**, *10*, e0120644. [CrossRef] [PubMed]
6. Liu, X.; Liu, Y.; Aberer, K.; Miao, C. Personalized point-of-interest recommendation by mining users' preference transition. In Proceedings of the 22nd ACM International Conference on Information & Knowledge Management, San Francisco, CA, USA, 27 October–1 November 2013; ACM Press: New York, NY, USA, 2013; pp. 733–738. [CrossRef]
7. Lian, D.; Zhao, C.; Xie, X.; Sun, G.; Chen, E.; Rui, Y. GeoMF: Joint geographical modeling and matrix factorization for point-of-interest recommendation. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014; ACM Press: New York, NY, USA, 2014; pp. 831–840. [CrossRef]

8. Li, X.; Cong, G.; Li, X.-L.; Pham, T.-A.N.; Krishnaswamy, S. Rank-GeoFM: A Ranking based Geographical Factorization Method for Point of Interest Recommendation. In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, 9–13 August 2015; ACM Press: New York, NY, USA, 2015; pp. 433–442. [CrossRef]
9. Cai, L.; Xu, J.; Liu, J.; Pei, T. Integrating spatial and temporal contexts into a factorization model for POI recommendation. *Int. J. Geogr. Inf. Sci.* **2018**, *32*, 524–546. [CrossRef]
10. Li, H.; Ge, Y.; Hong, R.; Zhu, H. Point-of-Interest Recommendations: Learning Potential Check-ins from Friends. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; ACM Press: New York, NY, USA, 2016; pp. 975–984. [CrossRef]
11. Gambs, S.; Killijian, M.-O.; Cortez, M.N.d.P. Next place prediction using mobility Markov chains. In Proceedings of the First Workshop on Measurement, Privacy, and Mobility, Bern, Switzerland, 10 April 2012; ACM Press: New York, NY, USA, 2012; pp. 1–6. [CrossRef]
12. Mathew, W.; Raposo, R.; Martins, B. Predicting future locations with hidden Markov models. In Proceedings of the 2012 ACM Conference on Ubiquitous Computing, Pittsburgh, PA, USA, 5–8 September 2012; ACM Press: New York, NY, USA, 2012; pp. 911–918. [CrossRef]
13. Cheng, C.; Yang, H.; Lyu, M.R.; King, I. *Where You Like to Go Next: Successive Point-of-Interest Recommendation*; AAAI Press: Washington, DC, USA, 2013.
14. Rendle, S.; Freudenthaler, C.; Schmidt-Thieme, L. Factorizing personalized Markov chains for next-basket recommendation. In Proceedings of the 19th International Conference on World Wide Web, Raleigh, NC, USA, 26–30 April 2010; ACM Press: New York, NY, USA, 2010; pp. 811–820. [CrossRef]
15. Liu, Q.; Wu, S.; Wang, L.; Tan, T. Predicting the Next Location: A Recurrent Model with Spatial and Temporal Contexts. *Proc. AAAI Conf. Artif. Intell.* **2016**, *30*, 194–200. [CrossRef]
16. Yao, D.; Zhang, C.; Huang, J.; Bi, J. SERM: A Recurrent Model for Next Location Prediction in Semantic Trajectories. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, Singapore, 6–10 November 2017; ACM Press: New York, NY, USA, 2017; pp. 2411–2414. [CrossRef]
17. Zhao, P.; Zhu, H.; Liu, Y.; Li, Z.; Xu, J.; Sheng, V.S. Where to Go Next: A Spatio-Temporal LSTM Model for Next POI Recommendation. *arXiv* **2018**. [CrossRef]
18. Baral, R.; Iyengar, S.S.; Li, T.; Balakrishnan, N. CLoSe: Contextualized Location Sequence Recommender. In Proceedings of the 12th ACM Conference on Recommender Systems, Vancouver, BC, Canada, 2 October 2018; ACM Press: New York, NY, USA, 2018; pp. 470–474. [CrossRef]
19. Chang, B.; Park, Y.; Park, D.; Kim, S.; Kang, J. Content-aware hierarchical point-of-interest embedding model for successive poi recommendation. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence Main Track, Stockholm, Sweden, 13–19 July 2018; Volume 20.
20. Liu, C.; Liu, J.; Xu, S.; Wang, J.; Liu, C.; Chen, T.; Jiang, T. A Spatiotemporal Dilated Convolutional Generative Network for Point-Of-Interest Recommendation. *ISPRS Int. J. Geo-Inf.* **2020**, *9*, 113. [CrossRef]
21. Guo, Q.; Sun, Z.; Zhang, J.; Theng, Y.-L. An Attentional Recurrent Neural Network for Personalized Next Location Recommendation. *Proc. AAAI Conf. Artif. Intell.* **2020**, *34*, 83–90. [CrossRef]
22. Qu, Z.; Xie, R.; Xiao, C.; Kang, Z.; Sun, X. The Elephant in the Room: Rethinking the Usage of Pre-trained Language Model in Sequential Recommendation. In Proceedings of the 18th ACM Conference on Recommender Systems, Bari, Italy, 14–18 October 2024; ACM Press: New York, NY, USA, 2024; pp. 53–62. [CrossRef]
23. Li, S.; Chen, W.; Wang, B.; Huang, C.; Yu, Y.; Dong, J. MCN4Rec: Multi-level Collaborative Neural Network for Next Location Recommendation. *ACM Trans. Inf. Syst.* **2024**, *42*, 1–26. [CrossRef]
24. Morzy, M. Mining Frequent Trajectories of Moving Objects for Location Prediction. In *Machine Learning and Data Mining in Pattern Recognition*; Perner, P., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2007; pp. 667–680. [CrossRef]
25. Cho, E.; Myers, S.A.; Leskovec, J. Friendship and mobility: User movement in location-based social networks. In Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, 24 August 2011; ACM Press: New York, NY, USA, 2011; pp. 1082–1090. [CrossRef]
26. Yuan, Q.; Zhang, W.; Zhang, C.; Geng, X.; Cong, G.; Han, J. PRED: Periodic Region Detection for Mobility Modeling of Social Media Users. In Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, Cambridge, UK, 11 August 2017; ACM Press: New York, NY, USA, 2017; pp. 263–272. [CrossRef]
27. Li, Z.; Han, J. Mining Periodicity from Dynamic and Incomplete Spatiotemporal Data. In *Data Mining and Knowledge Discovery for Big Data*; Chu, W.W., Ed.; Studies in Big Data; Springer: Berlin/Heidelberg, Germany, 2014; Volume 1, pp. 41–81. [CrossRef]

28. Chambers, J.A.; Mandic, D.P. (Eds.) *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures, and Stability*; Wiley Series in Adaptive and Learning Systems for Signal Processing, Communications, and Control; John Wiley: Chichester, NY, USA, 2010.
29. Covington, P.; Adams, J.; Sargin, E. Deep Neural Networks for YouTube Recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, 15–19 September 2016; ACM Press: New York, NY, USA, 2016; pp. 191–198. [CrossRef]
30. Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv* **2014**. [CrossRef]
31. Géron, A. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2nd ed.; O'Reilly Media, Inc.: Beijing, China; Sebastopol, CA, USA, 2019.
32. Jurafsky, D.; Martin, J.H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 2nd ed.; Prentice Hall: Upper Saddle River, NJ, USA, 2009.
33. Dymarski, P. (Ed.) *Hidden Markov Models: Theory and Applications*; IntechOpen: London, UK, 2011.
34. Rabiner, L.R. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* **1989**, *77*, 257–286. [CrossRef]
35. Łagowski, M. Rekomendacja miejsc z wykorzystaniem reguł sekwencyjnych, rekurencyjnych sieci neuronowych i danych przestrzennych z sieci społecznościowych. Master's Thesis, Institute of Computer Science, Warsaw University of Technology, Warsaw, Poland, 2019.
36. Pajączkowski, M.; Płachta, M. Mobilny system rekomendacji miejsc wykorzystyujący dane z sieci społecznościowych, łańcuchy Markowa oraz bieżącą historię miejsc odwiedzonych przez użytkownika. Master's Thesis, Institute of Computer Science, Warsaw University of Technology, Warsaw, Poland, 2019.
37. Enrico, P.; Giuseppe, R. Predicting Your Next Stop-over from Location-based Social Network Data with Recurrent Neural Networks. In Proceedings of the RecTour 2017, Como, Italy, 27 August 2017.
38. Levandoski, J.J.; Sarwat, M.; Eldawy, A.; Mokbel, M.F. LARS: A Location-Aware Recommender System. In Proceedings of the 2012 IEEE 28th International Conference on Data Engineering, Arlington, VA, USA, 1–5 April 2012; IEEE: New York, NY, USA, 2012; pp. 450–461. [CrossRef]
39. Wang, W.; Yin, H.; Sadiq, S.; Chen, L.; Xie, M.; Zhou, X. SPORE: A sequential personalized spatial item recommender system. In Proceedings of the 2016 IEEE 32nd International Conference on Data Engineering (ICDE), Helsinki, Finland, 16–20 May 2016; pp. 954–965. [CrossRef]
40. Ye, J.; Zhu, Z.; Cheng, H. What's Your Next Move: User Activity Prediction in Location-based Social Networks. In Proceedings of the 2013 SIAM International Conference on Data Mining, Society for Industrial and Applied Mathematics, Austin, TX, USA, 2–4 May 2013; pp. 171–179. [CrossRef]
41. Wang, F.; Meng, X.; Zhang, Y. Context-aware user preferences prediction on location-based social networks. *J. Intell. Inf. Syst.* **2019**, *53*, 51–67. [CrossRef]
42. Isinkaye, F.; Folajimi, Y.; Ojokoh, B. Recommendation systems: Principles, methods and evaluation. *Egypt. Inform. J.* **2015**, *16*, 261–273. [CrossRef]
43. Shani, G.; Gunawardana, A. Evaluating Recommendation Systems. In *Recommender Systems Handbook*; Ricci, F., Rokach, L., Shapira, B., Kantor, P.B., Eds.; Springer: Boston, MA, USA, 2011; pp. 257–297. [CrossRef]