공학석사학위논문

Selective Trajectory Memory Network and its application in Vehicle Destination Prediction

2019 년 2 월

서울대학교 대학원

산업공학과

Benjamin LEGER

Selective Trajectory Memory Network and its application in Vehicle Destination Prediction

지도교수 조 성 준

이 논문을 공학석사 학위논문으로 제출함

2018 년 12 월

서울대학교 대학원

산업공학과

Benjamin LEGER

Benjamin LEGER의 공학석사 학위논문을 인준함

위 원 장 Jonghun Park (인)

부위원장 Myung Hwan Yun (인)

위 원 Sungzoon Cho (인)

Abstract

Selective Trajectory Memory Network and its application in Vehicle Destination Prediction

Benjamin LEGER
Department of Industrial Engineering
The Graduate School
Seoul National University

Predicting efficiently the final destinations of moving vehicles can be of significant usefulness for several applications. Many probabilistic methods have been developed to address it but often include heavy feature engineering and do not generalize well to new datasets. To face these limitations, Deep-Learning models present the advantage of automating processing steps and can therefore be easily adapted to new input data. De Brébisson *et al.* proposed clustering based deep-learning approaches to solve it in the specific case of the prediction of Taxis destinations with remarkable performances, alongside with a proposition of a novel architecture inspired by Memory-Networks used in Natural Language Processing, and requiring no preliminary clustering. A large room for improvement was however left for the latter approach : the necessity of a relevant selection function retrieving historical trajectories similar to partial trips to predict was indeed outlined by the authors. In this work we propose to use the Segment-Path distance, introduced by Besse *et al.*

in former works on trajectory clustering, to come up with an improved architecture of this memory model. A review of several Memory Networks architecture and their applications in time-series prediction is provided to give an overview of the different structural alternatives existing for the design of our model architecture. Finally, our model is confronted to individual car data and we propose a personalized user-by-user prediction of destinations. We discuss the suitability and limits of the type of model in this specific problem and conclude that the promising obtained results are penalized by infrequent destinations cases inducing noise whose effect could be reduced by turning our approach into a classification problem.

Keywords: Destination Prediction, Retrieving function, Memory Networks, Segment-Path Distance, Car trajectories, User-personalized prediction

Student Number: 2017-29718

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivations, background

Monitoring and predicting locations and destinations of vehicles can be very useful for several purposes. First of all, it can allow a better anticipation of traffic and thus helping avoiding road congestion. For car manufacturers having real-time information on car user's location and directions can be used for marketing purposes such as targeted advertisements (e.g. recommendations of places to visit nearby current location), but it can also helps optimizing the control of vehicles (e.g. management of fuel consumption...).

GPS devices are often providing these positioning informations, but in the case where a driver is not using the built-in navigation system of his vehicle (for instance when following a well known itinerary, or using a personal GPS navigator system), no knowledge on his final destination is available. That is a reason why developping predictive models is necessary.

This task has been made easier by the growing availability of geolocation data provided by different types of mobile sensors which are increasingly used for modelling and analyzing the driving patterns of moving vehicles for the purposes we mentioned.

1.2 Problem Description : destination forecasting problem

1.2.1 General context

The corresponding general prediction problem consists in the determination of the final geographical coordinates $D = (d_{latitude}, d_{longitude}) \in \mathbb{R}^2$ of a moving vehicle given an initial portion (or "prefix") $P = (p_i)_{i \in \{1,\dots,n\}},\ p_i \in \mathbb{R}^2\ \forall i$ of its previous n coordinates and sometimes some context variables (metadata) describing the vehicle (e.g. driver ID) or the trip itself (e.g. departing time).

Taxis destination prediction is one of the main cases of application of this forecasting problem, with the notable example of the "ECML/PKDD 15: Taxi Trajectory Prediction (I)" challenge [1] . In this kind of framework, models are usually built on an important set of historical trajectories and associated metadata for taxis in a given region.

However the forecasting problem also applies to car manufacturers as they intend to provide user-personalized services based on driving destinations as mentioned in 1.1. This case belongs to what we will refer to as the *user-personalized destination prediction problem*. In this case, historical trajectories would tend to be collected individually for each user as they might intuitively present properties and patterns specific to each driver.

1.2.2 Specific problem tackled

The exact problem tackled in the 'Experiments' section of this paper (see section 4) has been issued by a famous Korean car manufacturer. This one provided individual

[1] https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i

driving data of ninety customers with the specific goal to derive models predicting their final destinations as accurately as possible, given partial completions of trips. The main constraint provided was the focus on deep-learning models as alternatives to models already used by the company.

Additionally to the recorded trajectories, four corresponding external variables about each trip were provided : the day of the week, the hour of the day, the week number (between 1 and 52) and a last one indicating if the trips were performed on a national holiday.

1.3 Existing models and methods

Several methods have been designed to solve the previous general problems. Bayesian inference framework is commonly used to derive the probability of a location to be the final destination of a given prefix. Ziebart *et al.* (22) and Krumm *et al.* (12) use a grid representation of the trajectory space to predict the probability of each node to be the final location of a given trip using Baye's rule, making use of additional external information such as elements about roads characteristics and driving behaviors. Xue *et al.* (19) proposed an improvement of these works in the case where no external data are available and where the historical trajectories are not sufficient to cover all possible routes that a vehicle can take ("*data sparsity problem*").

Several probabilistic methods based on Markov models has been developed to find the most likely future destination of moving vehicles. Among them, Alvarez-Garcia *et al.* (1) propose a Hidden Markov model generated from present location and past GPS recordings based on a reduced number of significant location points.

Krumm (12) uses a simple Markov model to predicts very next road segments visited by vehicles. Ashbrook *et al.* (2) propose a clustering based approach where Markov models are learnt to predict the most probable future destination among grouped frequently visited locations. As seen in these examples, both Bayesian and Markovian approaches are based, and thus strongly dependent, on a discretisation of the trajectory space.

Other works introduced alternative methods to retrieve the relevant information corresponding to a given prefix within a historical destination set, and to use it to derive a prediction model. Tiesyte and Jensen (16) developed a method to find the nearest neighbour of trajectories based on several distances measures in known roads and to use it to predict the future positions of public buses. Besse et al (4) introduced a metric for efficient trajectory clustering and used it in a later work (5) to propose a density based model where gaussian mixture models where fitted on the points of clusters of historical trajectory, allowing to compute similarities between new trajectories to predict and eventually give a prediction. Monreale *et al* (14) propose a pattern mining approach and build a decision tree to predict the very next location of a moving vehicle.

In order to design models requiring as few engineering features as possible, De Brebrisson *et al.* (9) proposed several artificial neural network architectures using GPS logs as well as external information (driver id, departure time ...) of taxi trips to accurately predict their destinations. One of the presented model ranked first at the "ECML/PKDD 15: Taxi Trajectory Prediction (I)" kaggle challenge. Lv *et al.* (13) modeles taxis trajectories as two-dimensional images and uses convolutional neural networks to extract spatial patterns used for prediction.

Regarding the exact problem introduced in 1.2.2, a deep-learning approach inspired by (9) has been presented by (23). The problem is however turned into a classification task were each trip is assigned to a class of final destinations clustered beforehand from historical trajectories.

1.4 Research Motivation and Contributions

In addition to its winning approach of the "ECML/PKDD 15: Taxi Trajectory Prediction (I)", (9) introduced a new type of model inspired by the mechanics of the "Memory models" used in the field of Natural Language processing, especially in question-answering tasks. This method was however more presented as a suggestion, introduced alongside with several directions for future improvements.

In this work, we propose to investigate in further details this type of architecture and to provide some improvements and alternatives in its general structure as well as for each of its components. The use of a distance measure between vehicle's trajectories introduced by (4) is especially considered to improve the selection of historical information by this neural network. More generally we try to provide a general understanding of this family of models and of their application in trajectory destination forecasting. Finally we intended to apply the studied architectures on a user-personalized prediction problem provided by a famous Korean manufacturer.

The main contributions of this work are :

(a) a review and comparison of existing memory models with a specific focus on their applications on time-series forecasting problems and destination prediction tasks

(b) the proposition of a modified architecture of memory networks for trajectory destination prediction

(c) the study of a user-personalized prediction problem via a regression approach and the experimentation of our memory model on the provided data

1.5 Organization of the Thesis

This report is composed of 5 chapters.

In Chapter 2 we propose a review of several memory models, detail their architectures and structures, define their common components, see their areas of application and compare them by presenting their analogies and structural differences. We also define the Segment Path Distance created in (4) and show its fitness in the task of selecting relevant completed historical trajectories from a partially completed trip. Previous works on the user-personalized dataset used in the experimental section of this study are also mentioned.

In Chapter 3 we propose a new possible architecture built from the idea developed in (9) including several new elements inspired by the models studied in Chapter 2, and implementing an SPD-based selection of historical information module.

Chapter 4 present some experiments carried out on a dataset provided by a famous Korean manufacturer. The data are first briefly described and analysed and our model is applied and tested on several different cases of study. The viability of using this type of approach on user-personalized prediction problems is also discussed.

Finally Chapter 5 gives some concluding remarks and expose future research directions.

Chapter 2

Related works

2.1 Artificial neural network models for trajectory prediction

2.1.1 Encoding and clustering approach

As mentioned in Section 1 artificial neural network architectures have been designed to solve the moving vehicle destination forecasting problem. State-of-the-art performances have been obtained by (9) in the "ECML/PKDD 15: Taxi Trajectory Prediction (I)" kaggle challenge using an architecture based on multi-layer perceptrons and clustering of historical destinations.

General structure :

This model follows a more general structure presented by the authors and summarized by Figure 2.1.

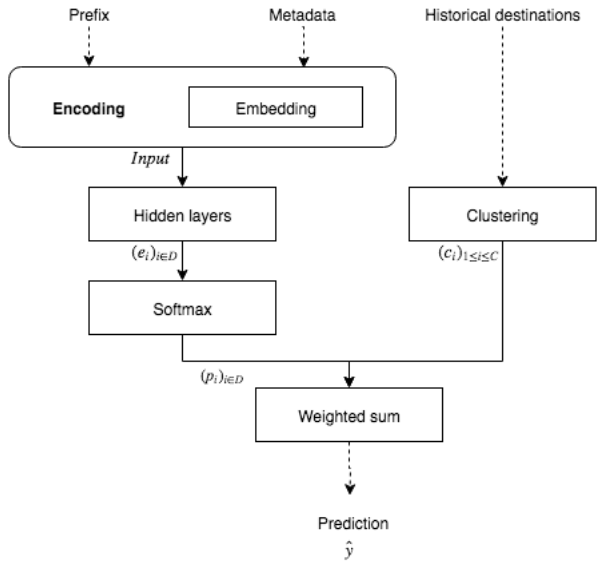

Figure 2.1: Diagram of the generic architecture presented introduced by de Brébisson *et al.* (9)

In this one the trajectory prefix to predict are first encoded into an internal representation $I(prefix)$ which is associated to a continuous representation of metadata (we consider n_{meta} categorical variable $(meta_i)_{i\in\{1,\dots,n_{meta}\}}$ of P_i possible values) obtained through embedding :

$$Input = [I(prefix), Embedding(meta_1), \dots, Embedding(meta_{n_{meta}})] \tag{2.1}$$

This input is then fed forward into several hidden layers whose final output (i.e. internal representation, noted $(e_i)_{i\in D}$, where D is a chosen hidden layer dimension) is eventually used to compute normalized weights through the use of the softmax activation function :

$$p_i = softmax(e_i) = \frac{exp(e_i)}{\sum_{j=1}^{C} exp(e_j)} \ \forall i \in D \tag{2.2}$$

These coefficients are finally used to weight centroids corresponding to clusters of historical destinations $(c_i)_{1 \leq i \leq C}$ (where C a chosen number of clusters) computed beforehand. This weighted sum is returned as a prediction :

$$\hat{y} = \sum_{i=1}^{C} p_i c_i \tag{2.3}$$

Encoding :

Two approaches to encode the initial prefix were proposed :

(a) ***multi-layer perceptrons (MLP) :***

In the former case, a fixed representation of the partial trajectory is beforehand computed were the first and last k GPS logs are extracted and concatenated, where $k \in \mathbb{N}$ to be set.

The reason for this stems from the very nature of multilayer perceptrons, allowing only fixed-length vectors as input were the trajectories of the training set presented various sizes.

Standard hidden layers were chosen (matrix multiplication, bias and application of a non-linearity function, chosen to be the Rectifier Linear Unit (ReLU)).

(b) ***recurrent neural networks (RNN) :***

In the case where RNNs were chosen to compute an internal representation of the input, long-term-short-term memory units (10) were used to process locally the inputted trajectories considering successively each pair of GPS coordinates and updating an internal state of a given encoding dimension.

The first main benefit of this approach lies in the fact that the Recurrent

Neural Network encoder allows the use of variable-length input vectors. In addition to that, the obtained encoding evaluates each prefix as a whole.

Alternatively a windowed version has been proposed, were the input of the RNN is set to be a succession of several successive GPS recordings (window) instead of individual points, for a better identification of short-term dependencies.

2.1.2 "Memory network" model for taxi trajectory prediction

Another related approach has been introduced by the same authors, inspired by mechanisms commonly used in the field of Natural Language Processing and formalized by works on Memory Networks ((18) (15)). Further insights on the general structure of this class of model will be given in section 2.2.

The proposed idea was to compare any new prefix $\mathcal{P}$ to predict to a set of historical trajectories $(\mathcal{T}_{ci})_i,\ i \in \{1, \ldots, m\}$ (the "memory" of the model) in order to compute a weighted prediction of already observed final destinations. The global architecture of the model is summarized by Figure 2.2.

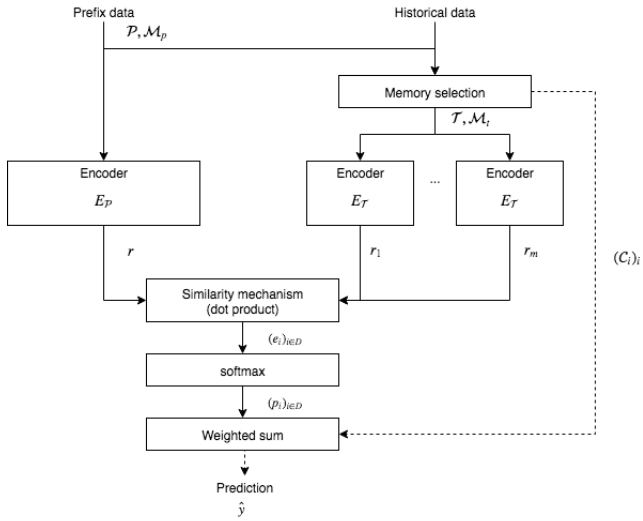

Figure 2.2: Diagram of the generic architecture presented in 2.1.2

Four main steps are considered :

Memory selection

m "candidates" historical trajectories $\mathcal{T}_{c_i}$ (called "candidates") are first selected within a wider pool of historical data. A random selection is implemented in (9) even though the authors suggest further investigations for finding an efficient selection method. The design of a retrieving function selecting the historical trips the more similar (given a certain criterion) to $\mathcal{P}$ is especially recommended.

Similarity ("attention") mechanism

In order to quantify the relative relevance with the prefix of each candidates in the memory , similarities are computed. A simple dot product is used by (9) but here again experimentations of more sophisticated similarity functions are suggested. A set of m similarity weights $(e_i)_i$, subsequently normalized using the softmax function,

are then obtained through this step :

$$e_i = \langle r, r_i \rangle, \ \forall 1 \leq i \leq m, \ \text{where } \langle ., . \rangle \text{ is the canonical scalar product in } \mathbb{R}^m \quad (2.4)$$

$$p_i = softmax(e_i) \quad (2.5)$$

Final prediction

A weighted sum of the final destinations $(\mathcal{C}_i)_i$ of the candidates is computed with respect to the similarity weights and returned as a prediction :

$$\hat{y} = \sum_{i=1}^{m} p_i \mathcal{C}_i \quad (2.6)$$

Possible improvements

As mentioned, several improvements and alternative to the model implemented by (9) have been left for future works and research.

In order to generalize the principles and mechanics used in that approach, the next section presents the original works on memory models.

2.2 Memory networks and applications

Works on memory networks as in (18) and (15) were the founding of the architecture presented in 2.1.2. In order to better apprehend the mechanics it uses, but also the different existing alternatives in its structure, we present here more generic details on this class of models, initially developed in the framework of Natural Language Processing (NLP). A direct application of the the end-to-end memory network of (15) in multi-dimensional time-series forecasting will also be presented.

2.2.1 MemNN models

Motivations

As mentioned, memory networks have first been introduced in Natural Language Processing and more specifically in the framework of question-answering tasks.

In this one, a story composed of several facts is typically provided alongside with a question related to it. Hence, this type of task requires from a model an efficient memorization of the mentioned information and an ability to use it in a relevant fashion. However, most of the language modelling models present serious limitations in performing it as pointed out in (21).

Principle

To overcome this, (18) proposed a type of model including a distinct memory component from which information could be easily written or read and used jointly with inputs for inference.

In opposition to commonly used RNN based models, which tend to compress past information and memory (through their internal representation, making it difficult to retrieve the information related to previously processed inputs), this approach intended to include a well compartimentalized memory component allowing it to accurately access and retrieve useful "memorized" inputs.

General structure

This memory, modelled by an indexed array $m = (m_i)_{i \in \{1,\dots,N_m\}}$ of a given size N_m, is working jointly with four distinct "components" described below :

(a) ***An input feature map :***

This part of the models is in charge of computing an internal representation of given inputs $x = (x_1, \ldots, x_n) \in E$, the input space. Typically an embedding model in the case of text inputs as considered in (18).

(b) ***A "generalization" module :***

The other main component of the model is related to the update of the memory based on the given inputs and their internal representation. In the simplest case new inputs are simply stored in empty memory slots without any modification of previous memory components. Hashing tricks are considered in (18) to select only a limited amount of relevant inputs to store in m.

(c) ***An output feature map :***

The inputs and the previously organised and updated memory are used jointly to compute an (internal) output. Typically similarities are computed between the input and each memory component. (18) consider a dot product based similarity function and select successively :

i. the most similar memory components to the input x (let us call it $m_{nearest_1}$)

ii. the most similar component ("$m_{nearest_2}$") to the joint embedding of the input and of $m_{nearest_1}$

and outputs the list $[x, m_{nearest_1}, m_{nearest_2}]$.

(d) ***A response :***

This "internal" output is then simply converted to a chosen format (e.g. an actual sentence or a word in a "question-answering" NLP task for instance)

However, the output feature map in the model proposed in this work includes the selection of the memory component of maximum similarity with the input. Hence, its training can not be directly performed from a simple "input-output" pair (the max and $argmax$ function being not differentiable) and requires a higher degree of supervision since information on the most relevant (similar) memory components ("supporting facts") has to be provided in the training set.

2.2.2 End-to-end memory networks (MemN2N)

To overcome this limitation, (15) proposed a "continuous" version of (18) that can be trained in an end-to-end fashion.

The core idea of this model was to weight the components stored in memory proportionally to their similarity to new inputs instead of only selecting the closest as done in (18), and thus obtaining a smooth and differentiable function mapping inputs and outputs, allowing back propagation and an end-to-end learning of the model's parameters.

We present here the general structure of this model, which can be seen as a modified special case of the architecture presented in 2.2.1.

Inputs

As described, this model can be trained from a simple pair of inputs and outputs a (a referring to the "answer" of a question-answering task as the one tackled in (18) and (15)).

The inputs are composed of :

(i) relevant information $in_1, \ldots, in_n$ to solve a given task (facts forming a story in the case of the considered example)

(ii) a query q

Memory selection ("generalization")

In this model the generalization step (as mentioned in 2.2) simply consists in splitting the inputs into two parts : storing $in_1, \ldots, in_n$ directly in memory and putting q aside.

Internal output computation

Two internal representations of the query (u, obtained through a first encoder E_B) and of the memory components (through an encoder E_A) are processed and dot product similarities are computed between the two resulting outputs. In a similar fashion as before these similarities are normalized using a softmax activation function and used to weight a representation of the memory. It has to be noted that this last representation could be obtained through a different embedding layer (E_C). The obtained output vector o is then added term by term with u.

Response computation

The previous sum is finally fed forward through a dense layer and the softmax activation function is used to select an output from a set of possible values (the considered example being a classification problem).

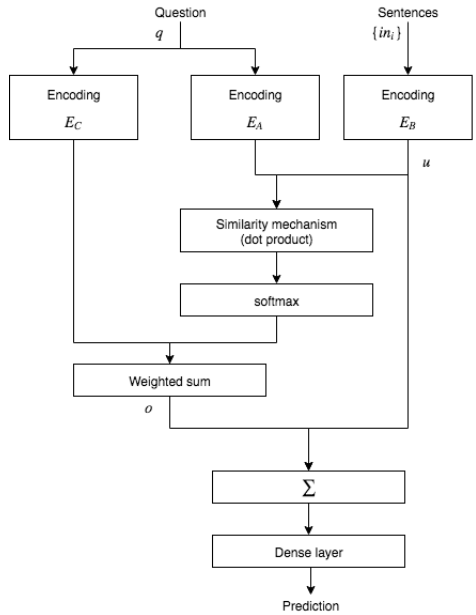

Figure 2.3: General structure of the end-to-end memory network of (15)

2.2.3 Memory networks for multi-dimensional time-series forecasting (MTNnet)

(6) proposed an application of the end-to-end memory network described in 2.2.2 to a problem closer to our interests in this study : multi-dimensional time-series forecasting.

Problem - Motivations

In that particular case the authors attempted to predict the future values of time-series $Y = \{y_1, \ldots, y_T\},\ y_t \in \mathbb{R}^D$ were T is the number of timestamps and D the dimension of the variable which is observed through time.

We describe here briefly this work as it tackles a task similar to our *destination prediction problem* : our car trajectories can indeed be seen as two-dimensional

$(D = 2)$ time-series.

Architecture

A similar mechanism as in 2.2.2 is used were $in_1, \ldots, in_n$ corresponds to historical time-series $X_1, \ldots, X_n$ associated to the studied variable and the question is replaced by a partially observed time-series Q. Encoding is performed through an architecture based on convolutional layers and Gated Recurrent Units (GRU) (8) for both Q and the historical series, and a filtering of the memory based on similarity weights obtained by taking the dot products $\langle Q, X_i \rangle, \ \forall i$ is performed in a similar fashion than in (15).

The main structural difference with the end-to-end memory network defined in (15) lies in the concatenation (instead of sum) between the selected output memory vector (refered to as o in 2.2.2) and the encoded query (the "question" q in 2.2.2, the partial time-serie here). A non-linear prediction y^D is then obtained by inputting the concatenation to a fully-connected layer.

This approach also includes an autoregressive component performing a linear prediction of the future values of Q and average with y^D, that will not be described in details here.

2.3 Analogies and comparisons between the memory models introduced

Five main families of models have been introduced in this section.

In all of them we can outline several important components of which we propose

the following 5-stage classification [1] :

Memory storage — How the inputs are stored in the memory components.

Encoding module — Module that computes an internal representation of the memory and of the query.

Memory selection mechanism — Given a memory and a query, how the model returns a "context vector" based of the relevant information related to the query to solve the task. It often includes

- a similarity mechanism — to compute similarity scores between memory elements and the query.
- a data selection process — which determines if the memory component should be used as a whole or only partially.

Query-memory combination — The mechanism used to combine the encoded query and the "context" memory vector.

Answer generation module — The structure used to return a prediction for a given task

Table 2.1 gives a comparison of the considered memory models based on these categories :

[1] This one is rather similar to the one given in (18) and presented in 2.2.1 but adds some new elements introduced by end-to-end architectures.

Table 2.1: Comparisons between the considered models based on the proposed classification of memory networks components

	MemN2N	**MTNet**	**Taxi memory network**
Memory storage	All inputs but query	All inputs but query	All inputs but query
Encoding	Word embedding	Convolution+GRU	[Metadata embedding + trajectory representation]
Memory selection	Continuous weighting	Continuous weighting	Continuous weighting
Similarity mechanism	Dot product	Dot product	Dot product
Data selection process	Total	Total	Partial (only destinations)
Query/memory combination	Sum	Concatenation	None
Answer	Dense layer	Dense layer	Weighted sum

The MemNN model have set the fundamental principles of memory networks with the introduction of a well compartimentalized memory component where information can be easily accessed to be combined with a query to infer a prediction. The MemN2N adapted this work to create and end-to-end memory network requiring less supervision : one of its key property was the continuous weighting of the memory to obtain a "context" vector combined with a representation of the query.

These models were initially developed for NLP question-answering tasks. However applications of these models have been made for time-series forecasting tasks. The MTNet took advantage of the MemM2N architecture, only introducing a concatenation between the context vector and the query instead of summing them.

The "memory network" of (9), used to solve a task very similar to ours, made

significant adjustments to the original MemM2N architecture : only the final destinations of the trips stored in memory were used to give a prediction and no combination between the prefix and the selected memory were made.

Figure 2.4 proposes a summary of the relationships between the model considered in this literature review.

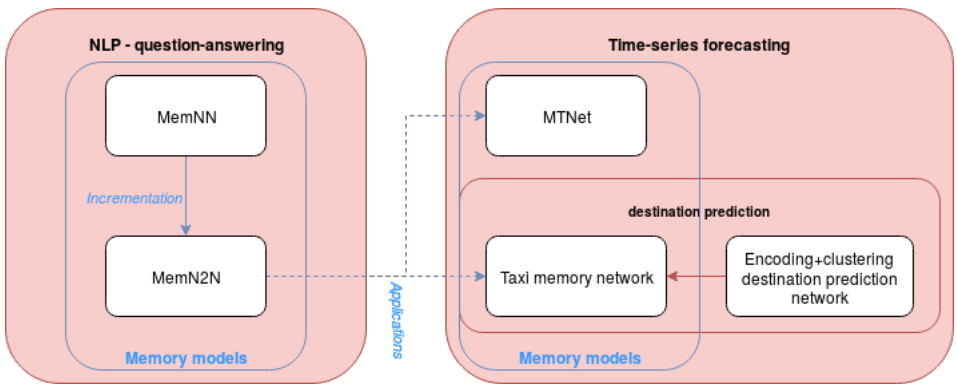

Figure 2.4: Relationships and comparisons between the model introduced in Section 2. Blue components correspond to model families and red components to problem types.

2.4 Distances measures for vehicle trajectories

As mentioned, the memory network architectures we considered might benefit from an efficient pre-selection of the relevant inputs to be stored in memory. This task can for instance be performed randomly (as done in (9), see 2.1.2) or through the use of hacking tricks (mentioned in (18)).

In the specific case of the *destination prediction problem*, (9) especially recommends the development of an efficient similarity measure to compare trajectories in order to only store the most similar to the prefix into memory.

More generally, the definition of a metric allowing a quantitative ranking of the

most similar historical trajectories to a partial trajectory to predict ("prefix") is an important asset in the type of task considered in this paper. More specifically we are here looking for a distance d that would ideally match the following criterion, were P is a "prefix" associated to a trajectory of actual final destination D, T_i, T_j full trajectories from a historical set H of respective destinations D_i and D_j and d_p a distance in $\mathbb{R}^2$ used to evaluate a prediction (typcially the euclidian distance):

$$d_p(D, D_i) \leq d_p(D, D_j) \iff d(P, T_i) \leq d(P, T_j),\ \forall (i, j) \in H \tag{2.7}$$

Based on a review of the numerous works on the topic ((17), (7), (20), (4) ...) we focus here on a relevant distance presented in the following subsection : the Segment-Path Distance.

2.4.1 Segment-Path Distance (SPD)

In a work initially intending to study the use of distances in the goal of performing clustering among two-dimensional trajectories, (4) introduced two distances : first the *"segment-path distance"* (SPD) used as a basis for the definition of the *"symmetric segment-path distance"* (SSPD) whose properties were described as excellent for clustering tasks.

Definitions

To define them rigorously we introduce first two preliminary distance :

Definition 2.1. *The Point-to-Segment distance between a point $p_1 \in \mathbb{R}^2$ and a segment s described by two points p_s^1 an p_s^2 is defined as follows :*

$$D_{ps}(p,s) = \begin{cases} \left\|p_1 p^{proj}\right\|_2 & \text{if } p_1^{proj} \in s \\ min(\left\|p_1 p_s^2\right\|_2, \left\|p_1 p_s^2\right\|_2) & \text{otherwise} \end{cases} \tag{2.8}$$

$$\text{Where } p_1^{proj} \text{ is the orthogonal projection of } p_1 \text{ in } s. \tag{2.9}$$

Definition 2.2. *The Point-To-Trajectory is defined as the minimum of all the Point-To-Segment distances between a point p_1 and each segment s_i composing a trajectory T^2 of size n_2 :*

$$D_{pt}(p_1, T^2) = min_{i \in [1,\dots,n_2]} D_{ps}(p_1, s_i) \tag{2.10}$$

Based on these distances the *segment-path distance* corresponds to the average of all the Point-to-Trajectory distances between each point p_i^1 of a trajectory T_1 of size n_1 and a trajectory T_2 :

$$D_{SPD}(T^1, T^2) = \frac{1}{n_1} \sum_{i=1}^{n_1} D_{pt}(p_i^1, T^2) \tag{2.11}$$

The SSPD distance is then simply defined as a symmetric version of this one :

$$D_{SSPD}(T^1, T^2) = \frac{D_{SPD}(T^1, T^2) + D_{SPD}(T^2, T^1)}{2} \tag{2.12}$$

However we focus our attention on the simple SPD distance, whose properties are discussed in the next subsection.

Properties and advantages in the historical trajectories selection task

A first interesting property of SPD and SSPD distances is their relative simplicity : they do not require any external parameter and are time insensitive, in opposition for instance to the Warping based distances (Dynamic Time Warping (3), Longest

Common Subsequence Distance (17) ...). They are also significantly less sensitive to noise than the latter family of distance, a significant comparative advantage when dealing with vehicle trajectories given the very noisy nature of this type of sequences.

Another interesting feature of the SPD distance for the task we intend to perform lies in the fact it compares both the geographical proximity and shapes of trajectories. As it can be noticed in its formulation (see 2.4.1) both the physical distance between two trips and their local geometrical similarities are taken into account. In our specific case we can easily imagine that two trips occurring in the same geographical area and following similar shapes (corresponding potentially to similar directions and itinerary) would intuitively end up in close final destinations.

Finally, the main advantage of the SPD distance in comparison to its symmetrized version is its ability to compare a partial trajectory with a fully completed trip, a property directly linked to its non-similarity. To illustrate that, we use the example of the Figure 2.5 In this one the trajectory T^1 is simply the beginning of the trajectory T^1 ($T^1 = T^2[:4]$). We have in this case $SPD(T^1, T^2) = 0 \neq SSPD(T^1, T^2)$.

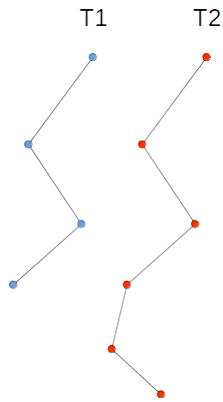

Figure 2.5: Example case where $SPD(T_1, T_2) = 0 \neq SSPD(T_1, T_2)$

The SPD distance will easily detect cases where a partial trajectory would correspond to the beginning of an previously recorded historical trajectory.

2.5 Personalized predictions on car manufacturer data

(23) has proposed an approach to solve the exact problem we tackled in this paper (1.2.2).

2.5.1 Problem approach and redefinition

In this one, trajectories from individual cars are used instead of taxi trips. The main implication of this lies in the fact that taxi usually visit very various destinations (dictated by the client) when car owners often follow driving routines (e.g. from home to work). One of the main challenge of user-personalized prediction is hence to take advantage of these patterns.

To take this into account (23) turned the initial destination prediction problem from a regression problem (predicting $\hat{y} \in \mathbb{R}^2$ as "close" as possible to the actual destination D) to a classification problem :

- clustering has first been performed on destination of a set of historical trip, resulting in several clusters of close destination;

- a model has then been derived to assign to each prefix one of these classes.

The prediction is said correct if it corresponds to the cluster whose centroid is the closest to the actual destination.

2.5.2 Method and model

A model similar to the one presented in 2.1.1 has been implemented assign one cluster of destination to each new partial trajectory to predict, the only difference being of course the use of a final softmax layer of size $n_{cluster}$, the number of destination clusters.

One model has been associated to each vehicle considered based on evaluations for different sets of parameters.

Chapter 3

Proposed Model

We propose here a new memory network architecture for trajectory destination prediction problems that make use of several elements studied in the previous section. Possible alternative to each element of our network are also given but not experimented.

3.1 Overall architecture

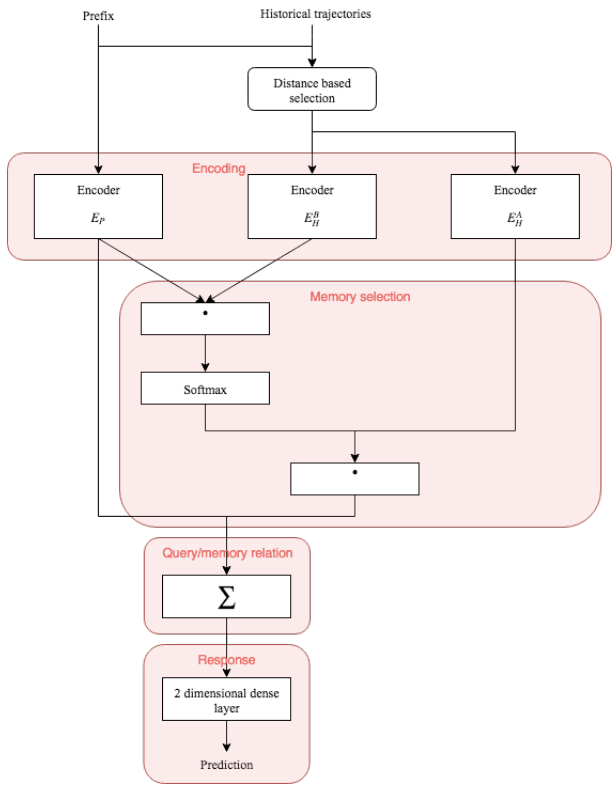

Figure 3.1: Overall architecture

Our model's general architecture in presented in Figure 3.1. We use a structure similar to Figure 2.3 adapted to trajectory data. It includes a continuous selection of memory component computed based on similarity weights. A regression model is considered as the main goal is to predict a two-dimensional (latitude, longitude) output. Hence a final two-dimensional fully connected layer is used to compute the prediction. If necessary in some particular cases, the network can easily be turned into a classification model as discussed in 3.7.

3.2 Input

For each instance the model is provided with :

(a) the partial trajectory to predict $p \in \mathcal{M}_{n_p \times 2}(\mathbb{R})$ and its associated n_{meta} metadata $meta_i,\ i \in \{1, \ldots, n_{meta}\}$

(b) a set H of historical trajectories.

3.3 Memory storage

The memory component of our network will be fed with a selection of the the m historical trips in H the more similar and close to each prefix p to predict. In order to "store" in memory trips that fit the best the criteria presented in 2.4 and selecting relevant trajectories we perform a full scan of H, computing the SPD distances between our partial trip and all the historical trajectories.

$$Memory\_Selection(H, p, m) = [T_{c_i} = T_i | T_i \in knn(p, H, SPD, m), T_i \in H] \quad (3.1)$$

$$\text{where } knn(x, E, d, m) \text{ returns the } m \text{ nearest-neighbours of } x \in E \quad (3.2)$$

$$\text{based on a distance } d \text{ defined on } E \quad (3.3)$$

We intend therefore to take advantage of the good properties of this Segment-Path Distance to improve the architecture proposed by (9).

3.4 Trajectory encoding

3.4.1 Encoding architecture

The trajectory encoding follows the first method introduced by (9) and presented in 2.1.1. As done in (23), the first and last $k = 5$ points of the (flattened) trajectories (prefix and historical) are concatenated to an embedded representation of

the metadata and fed-forward into one or several dense layers of dimension *Edim* (our encoding dimension). Recurrent Neural Networks could also have been used to encode our inputs, however no experimental results will be given with that type of architecture in this report.

3.4.2 Metadata and embedding

Three external variables are considered (the *day of the week*, the *hour of the day* and another indicating if the day is a national *holiday*). One embedding table is learned for each. The variable corresponding to the week number is however discarded.

Table 3.1 summarizes the original levels and the chosen embedding dimensions corresponding to each meta-variable.

Table 3.1: Original/embedding space dimension of the metadata

Metadata	Number of values/levels	Embedding dimension
Holiday	2	1
Week day	7	2
Hour of the day	24	4

3.4.3 Distinctions between encoders, weight-sharing

Three different encodings are performed : one for the prefix and two for the historical trajectories (see next section). Regarding those trips, same weights are shared between the different trips within each encoding module. However the two encoding (referred to as E_A^H and E_B^H) may have different weights $W_A^{enc} \neq W_B^{enc}$ (dense layer), and $W_A^{meta} \neq W_B^{meta}$, $\forall$ *meta* (metadata embedding) *(see formalization bellow)* :

Trajectory reshaping :

$$p^{resized} = concatenate(p[:5], p[5:]), \tag{3.4}$$

$$T_{c_i}^{resized} = concatenate(T_{c_i}[:5], T_{c_i}[5:]),\ \forall i \in \{1, \ldots, m\} \tag{3.5}$$

Metadata embeddings :

$$W_{meta}^{A}\delta_{meta}, W_{meta}^{B}\delta_{meta}, W_{meta}^{P}\delta_{meta},\ \forall meta \in \{holiday, weekday, hour\} \tag{3.6}$$

$$W_{meta}^{A}, W_{meta}^{B}, W_{meta}^{P} \in \mathcal{M}_{levels_{meta} \times EmbedDim_{meta}}(\mathbb{R}) \tag{3.7}$$

δ_{meta} being the one-hot-encoded vectors corresponding to each variable (3.8)

Encoding (case of one hidden layer) :

$$E_A^H = ReLU(W_A^{enc} concatenate(W_{meta}^{A}\delta_{meta}, T_{c_i}^{resized}, \forall i, meta)) \tag{3.9}$$

$$E_B^H = ReLU(W_B^{enc} concatenate(W_{meta}^{B}\delta_{meta}, T_{c_i}^{resized}, \forall i, meta)) \tag{3.10}$$

$$E_P^H = ReLU(W_P^{enc} concatenate(W_{meta}^{P}\delta_{meta}, p^{resized}, \forall i, meta)) \tag{3.11}$$

$$W_A^{enc}, W_B^{enc}, W_P^{enc} \in \mathcal{M}_{17 \times Edim}(\mathbb{R}) \tag{3.12}$$

$$E_A^H, E_B^H, E_P^H \in \mathbb{R}^{Edim} \tag{3.13}$$

3.5 Memory selection

The selection of relevant insights stored in memory are obtained through a process similar to the one introduced in 2.2.2.

3.5.1 Attention mechanism

A dot product is computed between the representation of the prefix (encoding E_P) and of the m candidates (obtained through the encoding E_A^H). The corresponding m weights are then normalized by the softmax function. A multiplication with the

second encoding of the memory (E_B^H) is finally performed to obtain output memory vector o :

$$p_i = \langle E_P, (E_A^H)_i \rangle \in \mathbb{R},\ \forall i \in \{1, \ldots, m\} \tag{3.14}$$

$$p^{norm} = softmax(p) \tag{3.15}$$

$$o = \sum_{i=1}^{m} p_i^{norm} (E_B^H)_i \in \mathbb{R}^{Edim} \tag{3.16}$$

3.5.2 Data used

In opposition to (9) the full candidates (and not only their final destinations) are used to infer the prediction. The main reason for that lies in the lower variability in the drivers destination : they obviously tend to follow more regular patterns than the taxis studied in this previous work. Hence a weighted sum of the candidates destinations would result in the prediction of a centroid not taking advantage of this property and probably in the middle of the frequent visited destinations. Depending on the cases of application, our model can however be easily adapted.

3.6 Query-memory association

We combine here the selected memory (corresponding to the output vector o) and the representation of the prefix E_P in an additive fashion :

$$a = o \oplus E_P \in \mathbb{R}^{Edim} \tag{3.17}$$

Concatenation, as done in (6), could also be considered as an alternative :

$$a = concatenate(o, E_P) \in \mathbb{R}^{2Edim} \tag{3.18}$$

3.7 Final prediction

The predicted destination is finally given after processing a into one last dense layer of dimension two :

$$\hat{y} = W^{out} a \in \mathbb{R}^2 \tag{3.19}$$

$$W^{out} \in \mathcal{M}_{Edim \times 2}(\mathbb{R}) \tag{3.20}$$

In the case where our model should be adapted to classification problem (as the one tackled by (23)) a softmax layer of the dimension n_{labels} could replace this ouput layer :

$$\hat{y} = softmax(W^{out} a) \in \mathbb{R}^{n_{labels}} \tag{3.21}$$

$$W^{out} \in \mathcal{M}_{Edim \times n_{labels}}(\mathbb{R}) \tag{3.22}$$

Chapter 4

Experiments

4.1 Objectives

Several experiments are carried out. They intend to :

1. provide insights of our user-personalized dataset and of its properties
2. evaluate the fitness of the proposed architecture on an example of user-personalized prediction problem

The model proposed in Section 3 is not *a priori* the most suitable kind of approach for that type of task. However we found interesting to study its performances and limitations in that context.

4.2 Dataset

We describe here in further details the dataset considered. The experiments carried out in this study make use of historical driving data from 90 vehicles (with a total initial number of 4,765,332 entry) provided by a famous car manufacturer. These driving data are composed of :

- (2 dimensional) GPS logs
- 4 external variables described in section 1.2

4.2.1 Variability and predictability

The main characteristic of the dataset lies in the high variability between user's data. The number of recorded trips, their length and duration as well as the covered geographical areas are especially very different from one vehicle to another and can greatly influence the performances of predictive models. Table 4.1 provides a summary of these properties.

Table 4.1: Statistics on vehicles characteristics (for minimum and maximum values the corresponding vehicle ID is indicated inside the brackets (e.g. : **(5)**))

	Nb. of trips	Avg. trip length	x-spread.	y-spread. .
Minimum	214 **(5)**	46.9 **(72)**	25.5 **(80)**	68.4 **(9)**
Maximum	2797 **(50)**	176.9 **(29)**	515 **(64)**	312.6 **(74)**
Average	1110	94.4	270.4	199.6
Std	437	26.6	92.47	64.4

The relevance of the fitting of a predictive model on each driver's dataset also highly depends on the structure of user's behavior. Two main critical cases are taken here into consideration :

(a) the drivers destinations are totally random. No structure can be extracted.

(b) the user's driving habits (and therefore the destinations he visits) changed dramatically over time.

In both cases the prediction of final destination can result in vain efforts. To quantify the "predictability" for each vehicle based on these considerations, we propose two metrics : the *Entropy* and the *Kullback–Leibler divergence*. The first of the two indicators quantifies the "unpredictability" and uncertainty of our data and can be formalised as follows :

Definition 4.1. *The Shannon Entropy of a discrete random variable D with values $\{d_i, 1 \leq i \leq n\}$ in a space of finite dimension n, of respective probabilities of occurrence $p_i = \mathbf{P}(D = d_i)$ is defined as :*

$$H(D) = -\mathbf{E}[log(\mathbf{P}(D))] = -\sum_{i=1}^{n} p_i log(p_i) \tag{4.1}$$

The more "unpredictable" the variable D is, the higher $H(D)$ will be.

Our target variable, the destination $D \in \mathbb{R}^2$, being continuous, a non-parametric estimation of the entropy is performed using the Kozachenko-Leonenko k-nearest neighbour estimator (11). This one computes an average of the distances between neighbouring data points of each observation to provide an estimation of the dispersion of the variable within an observation set.

The Kullback-Leibler divergence will be used to compare the evolution of the destinations distribution with time, in order to detect potentially important changes in driving behaviors. The definition of this dissimilarity measure is first given by :

Definition 4.2. *The Kullback-Leibler divergence of a discrete probability distribution P compared to a second probability distribution Q is given by :*

$$D_{KL}(P||Q) = \sum_{i=1}^{n} P(d_i) log\left(\frac{P(d_i)}{Q(d_i)}\right) \tag{4.2}$$

where $\Omega = \{d_i, 1 \leq i \leq n\}$ is the support of both probability distributions.

Once again a non parametric estimation is computed.

From this analysis some critical cases are noticed and discarded as discussed in the following section.

4.2.2 Considered vehicles

Among the 90 available vehicles, 20 are randomly selected for carrying out general experiments. The most unpredictable cases, based on the criteria discussed in 4.2.1,

are excluded before performing this selection. Quantitative analysis of the obtained results, especially regarding the influence of both models parameters and vehicle properties on predictions, will be performed on this set.

In order to give more qualitative and intuitive insights of the results, individual "case study" will also be carried out. For reasons of readability 5 vehicles are drawn at random and will be briefly individually analysed. Among them we intentionally choose 4 "normal" cases, extracted from the set of 20 cars previously mentioned, and add a more "problematic" case, particularly complicated to predict. A quick description of the corresponding extracted dataset is proposed in Table 4.2.

Table 4.2: Summary of the 5 example vehicles properties (Averaged trip length in number of points). Car "5" corresponds to the more difficult case. Averages are computed for the first four cars.

Vehicle ID	Nb. of trips	Avg. trip length	x-spread.	y-spread.	Est. Entropy
4	647	79.72	295.3	195.4	0.090
16	1408	67.69	215.3	106.6	0.030
22	1571	80.76	298.6	219.3	0.082
44	1308	84.5	471.8	75.5	0.076
5	214	176.4	161.0	110.1	0.232
Average	1234	78.16	320.3	149.2	0.070

The four first vehicles seem to be relatively easy to predict, as they have covered relatively short trips on relatively restricted geographical areas, and as well provide a reasonable amount of data. In the other hand, with very long trips, few available data, and a relatively high estimated entropy, car "5" properties are expected to make the prediction task significantly harder.

Figure 4.1 gives the available trajectories for these 5 "case study" examples.

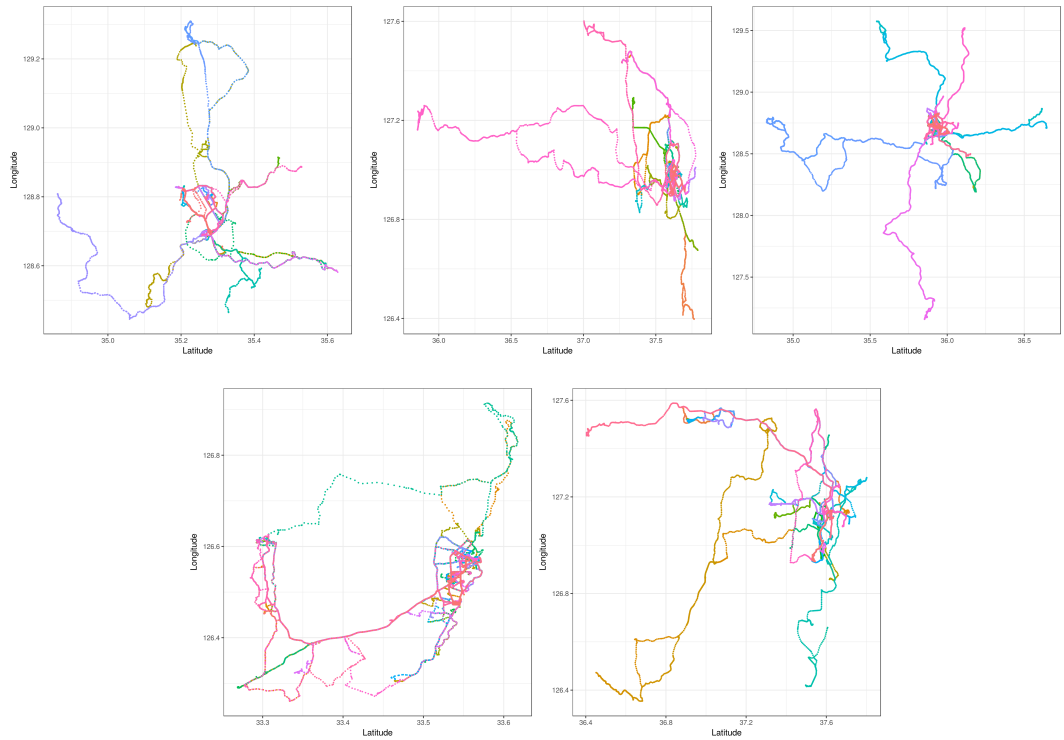

Figure 4.1: Trajectories of vehicles (respectively from left to right and top to bottom) 4, 16, 22, 44 and 5.

4.3 Experimental settings

4.3.1 Training and testing set

Our model is trained and tested for the selected vehicles. A 80%/20% (respectively for training and testing data) split of the data is performed for each car. The delimitation of pools of historical trajectories within which the memory of our models will draw its inputs is performed following two procedures, distinguishing the cases of the training and of the testing sets :

(a) *For the training set :* the m-closest trajectories (in the SPD sense) to each partial trip to predict are selected **within the training set excluding (of**

course) the prefix itself

(b) *For the testing set :* the candidates are selected **within the whole training set**

One testing/training set is considered per pair $(vehicle, completion)$ with $completion \in [5mn, 10mn, 20mn]$.

4.3.2 Test methodology and parameters

In order to associate a predictive model to each vehicle we propose the simple following strategy : several combinations of parameters are used to train multiple models for each pair *(user, completion)* and are then evaluated on the corresponding testing sets. The model with the best performances is then simply associated to the considered configuration (in short, one model per *(vehicle, completion)* couple).

Two main elements need then to be defined :

(a) a grid of parameters to be tested

(b) an evaluation metric to assess the performance of each model

Parameters

The grids of values considered for each parameter of our model are summarized in Table 4.3.

More detailed grids of parameters should be considered for a good and precise choice of model setting per vehicle, however as the goal of this experiment is mostly to study qualitatively the influence of these parameters on each type of vehicle and to assess the behavior of our model in this context, relatively restricted number of possible values are tested. A finer tuning is here left as a potential future work.

Table 4.3: Model's parameters

Parameters	Values
Encoding	-
Encoder type	MLP and embedding
Hidden layer structure	**-**
Number of nodes	[20, 50, 100]
Number of hidden layers	[1, 2, 3]
Activation function	ReLU
"Historical selection" parameters	**-**
Percentage of training selected	[0.05, 0.1, 0.3]
Hyper-parameters	**-**
Learning rate	0.001
Optimizer	Adam
Validation size (early stopping)	20%

Metric

To define an error metric we first need an efficient quantification of the distance between two locations. In order to take into account the curvature of the earth when measuring the distance between two points, we consider the distance defined by Definition 4.3.

Definition 4.3. *We define our corrected distance between two points* $A,\ B \in \mathbb{R}^2$ *as*

$$d_c(A, B) = \sqrt{111.0(A_x - B_x)^2 + 88.8(A_y - B_y)^2} \tag{4.3}$$

$$A = (A_x, A_y),\ B = (B_x, B_y) \in \mathbb{R}^2 \tag{4.4}$$

The two coefficients (110.0 and 88.8) being derived from the latitude of Korea ($37°$*).*

Predictions are then evaluated by comparing the averaged corrected distance as detailed by Definition 4.4

Definition 4.4. *We define our error metric as the averaged error distance between predictions* $\hat{y}_i$ *and actual destinations* D_i, $i \in \{1, \ldots, n\}$, *where* n *is the size of the*

evaluation set and d_c the distance introduced in 4.3, as :

$$Err = \sum_{i \in \{1,\dots,n\}} \frac{d_c(D, \hat{y})}{n} \tag{4.5}$$

4.3.3 Baseline model : simple encoding

In order to obtain a model to compare our selective memory network with, we consider here a baseline approach corresponding to a regression version of the model used by (23).

In this one we simply consider our basic encoder structure taking each partial destination to predict and the corresponding metadata as inputs, and simply add a final two-dimensional dense layer to predict each final destination.

The idea is here to assess the added value of the memory structure of our proposed approach to solve our prediction task, our baseline model basically being a memoryless version of it.

4.4 Experimental results

4.4.1 General results

The results per vehicle and per completion level are given by Figure 4.2. Only the best (selected) models for each pair $(car, completion)$ are considered.

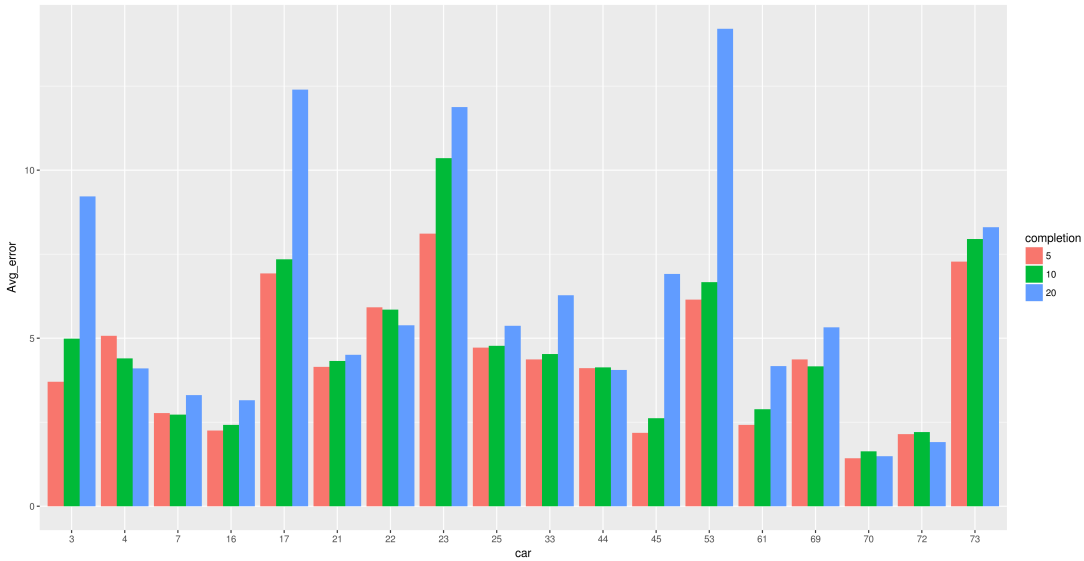

Figure 4.2: Averaged prediction error (in kilometers) per vehicle per completion level.

We first have a look at the extreme cases with the best and worst vehicles, respectively car 70 and 23.

Table 4.4: Detailed test results for the best and worst car.

Configurations		**Best models**			
		Parameters			***Error distance (km)***
vehicle ID	Compl.	Hist. Prop	Enc. Dim.	#hid. layers	
70	5	0.1	20	2	0.89
	10	0.05	100	3	1.12
	20	0.3	20	1	1.13
23	5	0.3	50	2	7.22
	10	0.1	100	3	9.30
	20	0.3	20	2	10.51

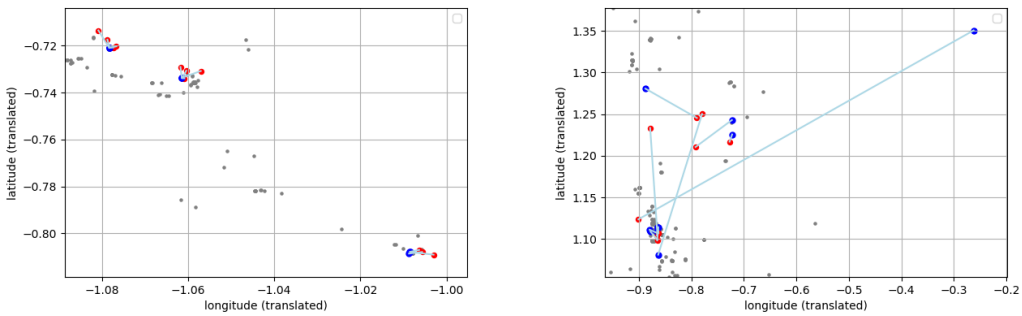

Figure 4.3: 10 randomly drawn predictions (red) vs actual destinations (blue) for vehicle 70 (best, left) and 23 (worst, right) at completion 5mn. The historical destinations of each car are drawn as grey dots.

In the case of car 70 the distribution of the final destinations is relatively ordered and we can distinguish four main areas the driver frequently visits. This vehicle also has relatively standard properties, with trip lengths and a data size close to the overall mean values. Our Selective Memory Model catches here very well the region towards which the vehicle is heading to and predictions are very satisfying.

Vehicle 23 also presents rather standard characteristics but with a less ordered repartition of the final destinations. The mean prediction error seems here to be penalized by one case where an infrequent and far away location is visited by the user. As it will be discussed later on, a classification approach would here help us to reduce the error by reducing the "noise" induced by this type of "outlier" destination.

One other main observation that can be done regards the surprisingly high errors for higher completions, but this one will be tackled in the next subsection.

4.4.2 Factors of influence on models performances

We study here briefly different factors that could impact the prediction results. We divide them into two main categories :

(a) the characteristics of the vehicles themselves : the size of training sets, the lengths of trips, the size of geographical area covered and the estimated entropy

(b) the parameters of the model : the complexity of encoding (number of hidden layers and encoding dimension) and the number of historical trajectories stored in memory

Vehicle properties

One first intuitive factor that would influence the prediction precision would be the length of trips. Figure 4.4 gives the averaged error distances by vehicles averaged trip's length.

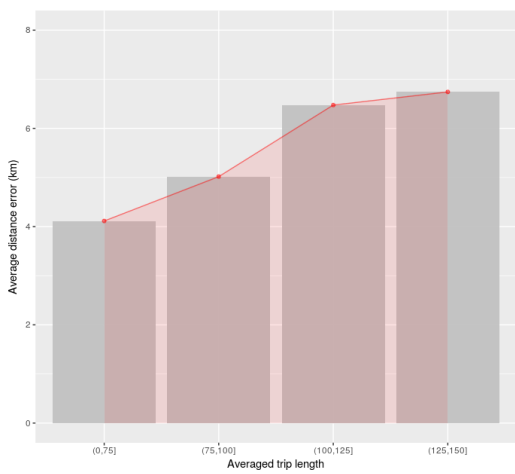

Figure 4.4: Averaged error (in kilometer) per vehicle averaged trip length (intervals of 20).

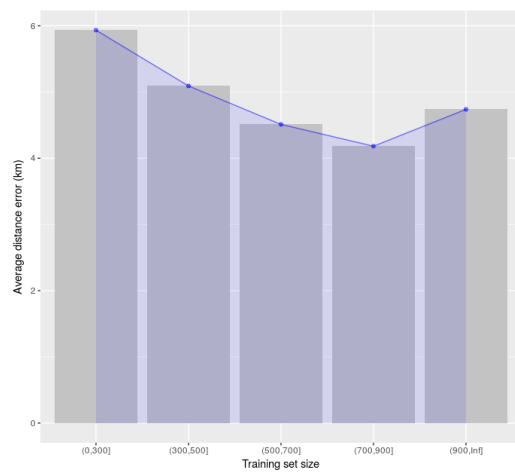

Figure 4.5: Averaged error (in kilometer) per vehicle averaged training set size (intervals of 300).

From this plot we can distinguish a relatively obvious correlation between the error and the averaged length which is confirmed by a non-parametric Pearson correlation test. Based on this one the correlation between the two variables is considered to be statistically significant at a level of 0.01.

Regarding the number of training data, Figure 4.5 tends to show a relatively clear influence of the size of training sets on the prediction precision, even though the error seems to increase in the case of the larger datasets. This can be explained by the fact that vehicles with very high numbers of recorded trips tend to have more chaotic destinations distribution (noise can be added by trips performed only once or twice for instance, as seen in the case of vehicle 23).

We also observe that fewer data are available for higher levels of completion (Figure 4.6). This phenomenon can be easily explained by the fact that trajectories shorter than each given completion (for instance trajectories lasting less than 20 minutes for a completion level of 20) are obviously discarded during preprocessing. This is here considered as one of the main explanation for higher prediction errors for high levels of completion in several vehicle cases.

As expected higher entropy in vehicles destinations distribution results in poorer accuracy as can be seen in Figure 4.7.

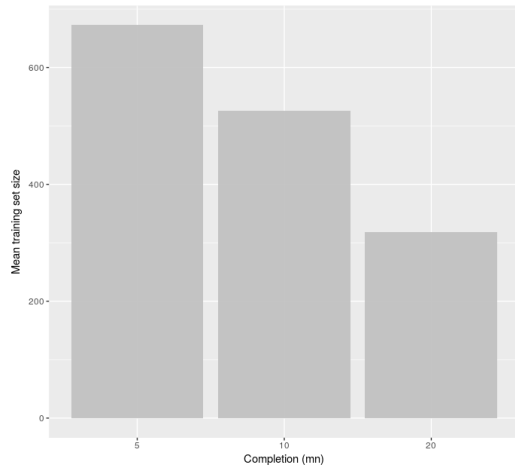

Figure 4.6: Averaged training sizes per completion levels.

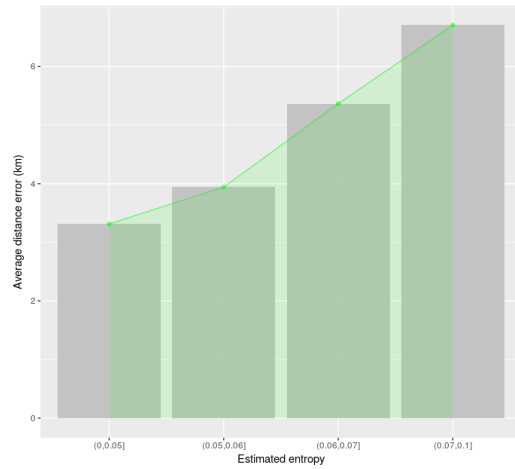

Figure 4.7: Averaged error (in kilometer) per estimated entropy (intervals of 0.01).

Models parameters

We now have a quick look at the influence of the models parameters in the final averaged prediction error. Figure 4.8 gives the the mean distance between our predictions and the actual destination for the different levels of complexity of our encoders. We took both the encoding dimension and the number of hidden layers into consideration.

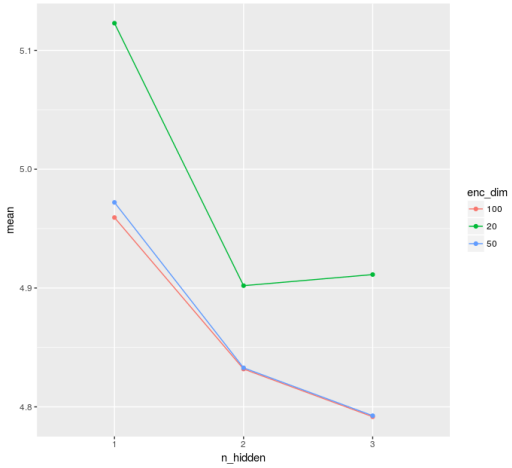

Figure 4.8: Influence of both encoding dimension and number of hidden layers in averaged error distances for all 20 vehicles.

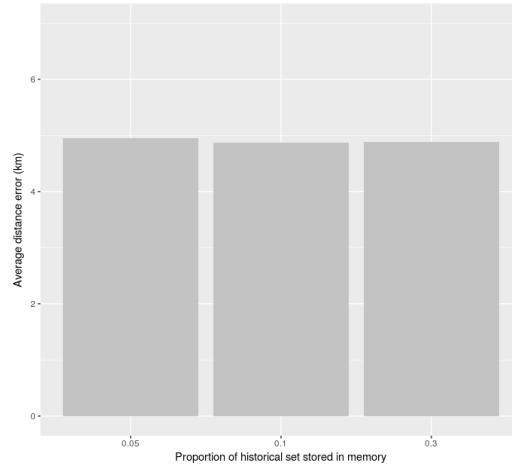

Figure 4.9: Influence of the proportion of the historical set stored in memory in averaged error distances for all 20 vehicles.

Our experimental results tend to show that more complex encoding modules result in better predictions, even though the difference between each encoding architecture seems to be relatively small and possibly non significant.

Regarding the number of historical trajectories stored in memory (more specifically the proportion of stored trips), the general influence on the prediction error does not seem to be significant in the overall results (grouping the 20 vehicles all together) as can noticed in Figure 4.9 (a very slight tendency for better predictions with greater proportions is however noticeable, but probably not significant). The setting of this parameter is dependant on each vehicle's properties.

The low numbers of parameters considered in our experimental setting does not allow us to very efficiently catch the influence of the model's characteristics on the obtained error. A study in more detailed parameters grid would be interesting in potential future works on this topic. It must be added that an individual analysis of

each vehicle should in practice be carried out, but is not proposed here in each car's cases for obvious reasons of clarity and readability.

4.4.3 Case studies : 5 example vehicles analysis

Table 4.5 summarizes the "best" models for each configuration *(vehicle, completion)* in the sense of the error metric introduced in Definition 4.4 for each one of the 5 selected example vehicles. In each case the corresponding averaged error distance as well as the associated set of parameters are detailed.

Table 4.5: Test results for the five example vehicles.

Configurations		**Best models**			
		Parameters			***Error distance (km)***
vehicle ID	Compl.	Hist. Prop	Enc. Dim.	#hid. layers	
4	5	0.1	50	2	4.70
	10	0.3	20	2	3.56
	20	0.1	20	2	3.43
16	5	0.3	50	3	2.02
	10	0.3	20	1	2.19
	20	0.05	100	3	2.68
22	5	0.3	20	1	5.51
	10	0.05	100	1	5.52
	20	0.05	50	2	5.06
44	5	0.3	20	3	3.82
	10	0.3	20	3	3.79
	20	0.3	20	3	3.79
5	5	0.05	50	2	16.9
	10	0.1	50	3	17.2
	20	0.1	50	1	16.8

Relatively decent precision seems to be obtained for three of the first four "normal" vehicles (4, 16 and 44). Experiments on car "22" seem however to result poorer

predictions. As expected vehicle 5 reveals to be considerably more difficult to predict with average errors above 15 kilometers.

Parameters settings tend to be relatively similar for the different completion levels for each vehicle.

Property of predictions : an example

To have a better insight on the property of the models in each specific case, but also to try understanding their performances, we have a closer look at the predictions. Figure 4.10 gives the actual destinations and the corresponding predictions made by the selected models in the case of vehicle 4 (rather "good" predictions), 22 (average performances) and 5 (poor precision), with a ten minutes completion. Only the ten first instances of each testing set are considered for readability.

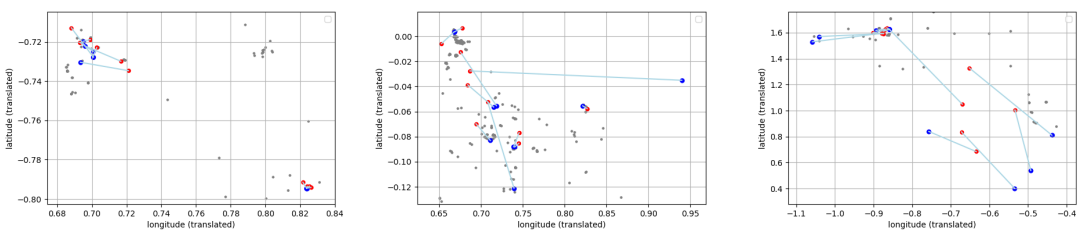

Figure 4.10: 10 first predictions (red) vs actual destinations (blue) for vehicles 4, 22 and 5 (from left to right), with a completion of ten minutes. Distances are plotted in light blue and destination of the training set in grey.

vehicle 4

Three distinct groups of destinations can be distinguished for that vehicle. Predictions are then relatively easy and the model performs quite well. A classification model assigning each prefix a class among clusters of historical trajectories, as done in (23), would be especially relevant in that specific case. However our regression

model tends to already return predictions corresponding to the correct geographical area.

vehicle 22

Predictions for vehicle 22 tend to be penalized by infrequent destinations. The example of the point at the extreme right of the plot in Figure 4.10 shows how some very far away and occasional locations visited by the user can have a negative influence on averaged prediction performances. Moreover the wide geographical area covered by this car (in comparisons to the other considered vehicles) is another simple factor that makes predictions error more important in this case.

vehicle 5

As expected vehicle 5's destinations prediction was a particularly difficult task given its properties : almost no routines can be noticed for this user, the trips covered are long (in a wide geographical area) and the amount of available data is low.

If the distances between predictions and actual destinations tend to be significant, the example of Figure 4.10 shows how our model manages to capture tendencies on the general direction the driver is heading to, based on each partial trip.

The same general properties defined in this section are noticed when considering other completion levels.

4.4.4 Baseline model

Our baseline model is then finally evaluated in the same testing sets and its results are compared to those obtained for our selective memory network. Figure

4.12 gives the difference between the prediction error for the best parameter settings of the baseline approach and of our memory model ($Error(memory\ model) - Error(baseline\ model)$).

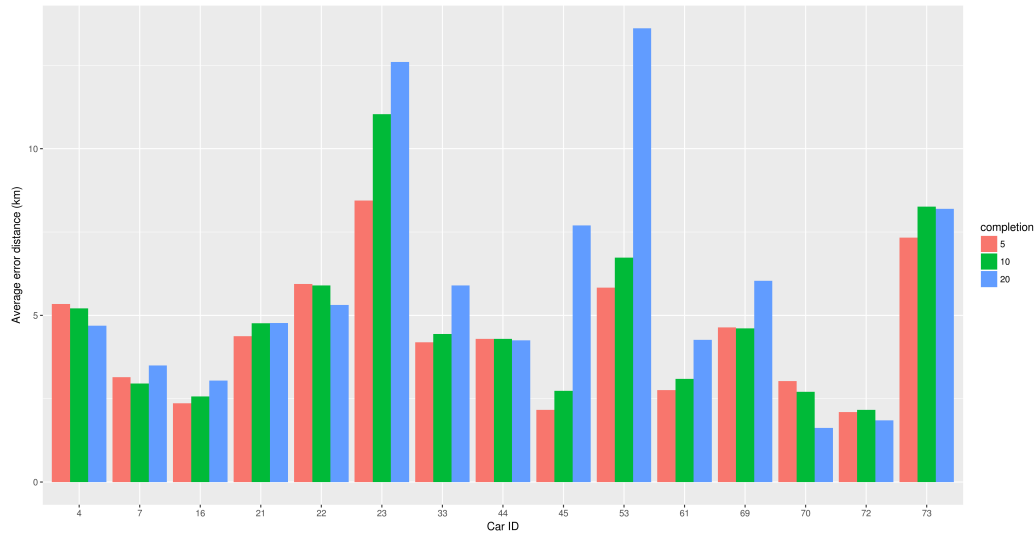

Figure 4.11: Averaged prediction error (in kilometers) per vehicle per completion for the baseline model.

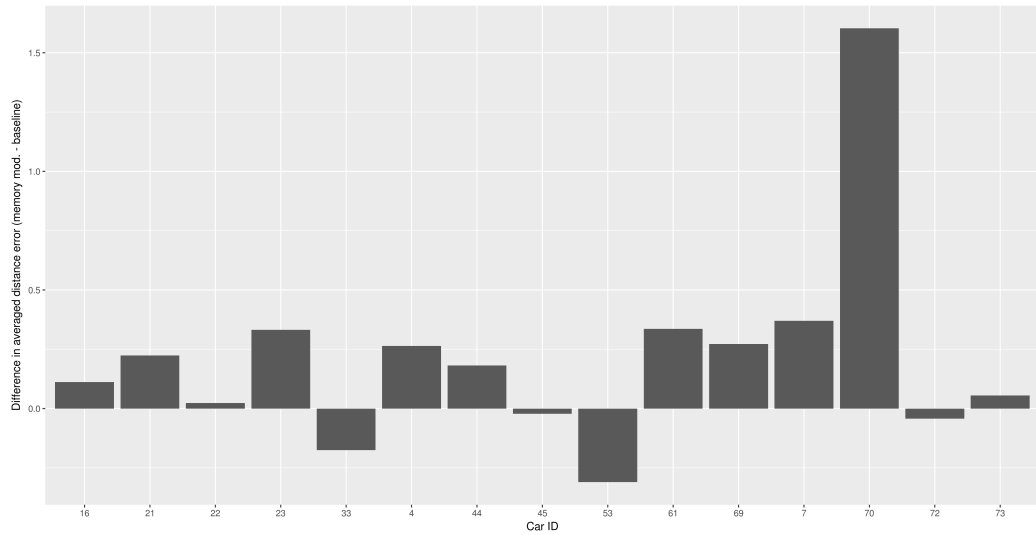

Figure 4.12: Differences of averaged prediction error (in kilometers) between the baseline approach and the memory model (per vehicle).

Better prediction results are observed in most cases for our memory model however the differences are most of the time relatively small. Vehicle 70 is the one which tends to benefit the most of the memory mechanisms included in our proposed approach. Conversely, cars 53, 33 and 72 predictions are more accurate with a simple encoding of the input data. These three vehicles share the same characteristic of performing short trips in restricted geographical areas, with a large set of available historical data. It could then be deduced that the simple baseline model would be more suited to these "easier" and sometimes almost trivial cases of very local driving behaviors.

4.5 Discussions

The experimental results presented in this section has outlined some relatively interesting properties of our selective memory network approach in this example of user-personalized prediction problem. Relatively good predictions have indeed been achieved.

It has been shown that the memory component of our model improved the prediction precision in almost all considered cases in comparison to the simpler neural encoding approach used by (23) that we adapted to regression. However less complex models must be preferred in some cases, especially when drivers are performing small trips in small geographical regions (corresponding to "easier" prediction problems).

Moreover, the type of regression approach implemented here is probably not the best suited for the data considered in this experimental section. A classification formulation of our problem, as the one tackled in (23) would probably have more sense in this context. Based on the destinations distribution for most of the vehicles it would be more interesting and sufficient for a car manufacturer to try predicting general areas toward which a driver would be heading to without attempting to precisely forecast exact destinations. The numerous infrequent historical visited locations observed in most cases could indeed be seen as noise, and it might be wiser not to take them into account as they would influence negatively the prediction of frequent cases which are predominant in this framework, where most of the users follow relatively strong routines.

A classification version of our model could then be easily designed and tested (minor modifications in the output layer and clustering of historical destination would probably be sufficient to achieve that) for this kind of user-personalized tasks.

More model's parameters settings must as well be tried out for a better understanding of their influence in prediction results. Baseline models of more traditional probabilistic methods should also be implemented on these data in order to perform a relevant comparison of performances.

Chapter 5

Conclusion

5.1 Conclusion

In this study we have given an overview of the application of memory networks in destination prediction problems for moving vehicle. In addition to discussing the possible architecture these model could take in this context we have introduced the use of the Symmetric-Path Distance in the pre-selection of historical data into memory.

Based on that we have proposed a model, alongside with several alternative for each of its components, summarizing the practices in use in different fields of application, from Question-Answering tasks in Natural Language Processing to multi-dimensional forecasting. This one can be easily modified depending on the exact type of task to perform.

We also introduced an example of user-personalized destination prediction problem, were we described individual datasets with very varying characteristics and used it as an example to implement our model with several parameters settings. Even though the choice of a regression model was probably not the most appropriate in this context, we observed some decent predictions in most of the studied cases, which outperformed the simple encoding model used in (23). However the

classification approach considered in that work was still probably more relevant and appropriate in this type of problem.

5.2 Future Directions

The experiments and tests done for our model in this work are not sufficient to correctly assess our model's performances and to conclude about his suitability in this specific context. More parameters should be tried out and more vehicles considered. Implementing more "baselines" approaches based on state-of-the-art methods on the data would also be useful to benchmark the performances.

More importantly a regression version of our Selective Memory Model should be implemented and tested in this context. As it outperformed the regression version of the encoder model used in (23) it would be interesting to see if it could obtained better accuracy in a classification task.

Regarding application in non-personalized problems, the public dataset of the "ECML/PKDD 15: Taxi Trajectory Prediction (I)" kaggle challenge could also be used for evaluation, especially in order to see if our SPD based pre-selection module is improving the results obtained by (9) in similar conditions.

Bibliography

[1] J. Alvarez-Garcia, J. Ortega, L. Gonzalez-Abril, and F. Velasco, *Trip destination prediction based on past gps log using a hidden markov model*, Expert Systems with Applications, 37 (2010), pp. 8166 – 8171.

[2] D. Ashbrook and T. Starner, *Using gps to learn significant locations and predict movement across multiple users*, Personal Ubiquitous Comput., 7 (2003), pp. 275–286.

[3] D. J. Berndt and J. Clifford, *Using dynamic time warping to find patterns in time series*, in Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining, AAAIWS'94, AAAI Press, 1994, pp. 359–370.

[4] P. Besse, B. Guillouet, J.-M. Loubes, and F. Royer, *Review & Perspective for Distance Based Clustering of Vehicle Trajectories*, IEEE Transactions on Intelligent Transportation Systems, 17 (2016), pp. 3306–3317.

[5] P. C. Besse, B. Guillouet, J. Loubes, and F. Royer, *Destination prediction by trajectory distribution-based model*, IEEE Transactions on Intelligent Transportation Systems, 19 (2018), pp. 2470–2481.

[6] Y.-Y. Chang, F.-Y. Sun, Y.-H. Wu, and S.-D. Lin, *A memory-network based solution for multivariate time-series forecasting*, 2018.

[7] L. Chen, M. T. Özsu, and V. Oria, *Robust and fast similarity search for moving object trajectories*, in Proceedings of the 2005 ACM SIGMOD international conference on Management of data, ACM, 2005, pp. 491–502.

[8] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, *Empirical evaluation of gated recurrent neural networks on sequence modeling*, CoRR, abs/1412.3555 (2014).

[9] A. de Brébisson, É. Simon, A. Auvolat, P. Vincent, and Y. Bengio, *Artificial Neural Networks Applied to Taxi Destination Prediction*, ArXiv e-prints, (2015).

[10] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, Neural Computation, 9 (1997), pp. 1735–1780.

[11] L. F. Kozachenko and N. N. Leonenko, *Sample estimate of the entropy of a random vector.*, Probl. Inf. Transm., 23 (1987), pp. 95–101.

[12] J. Krumm and E. Horvitz, *Predestination: Inferring destinations from partial trajectories*, in UbiComp 2006: Ubiquitous Computing, P. Dourish and A. Friday, eds., Berlin, Heidelberg, 2006, Springer Berlin Heidelberg, pp. 243–260.

[13] J. Lv, Q. Li, and X. Wang, *T-CONV: A convolutional neural network for multi-scale taxi trajectory prediction*, CoRR, abs/1611.07635 (2016).

[14] A. Monreale, F. Pinelli, R. Trasarti, and F. Giannotti, *Wherenext: A location predictor on trajectory pattern mining*, in Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '09, New York, NY, USA, 2009, ACM, pp. 637–646.

[15] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus, *Weakly supervised memory networks*, CoRR, abs/1503.08895 (2015).

[16] D. Tiesyte and C. S. Jensen, *Similarity-based prediction of travel times for vehicles traveling on known routes*, in Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS '08, New York, NY, USA, 2008, ACM, pp. 14:1–14:10.

[17] M. Vlachos, G. Kollios, and D. Gunopulos, *Discovering similar multidimensional trajectories*, in Data Engineering, 2002. Proceedings. 18th International Conference on, IEEE, 2002, pp. 673–684.

[18] J. Weston, S. Chopra, and A. Bordes, *Memory networks*, CoRR, abs/1410.3916 (2014).

[19] A. Y. xue, R. Zhang, Y. Zheng, X. Xie, J. Huang, and Z. Xu, *Destination prediction by sub-trajectory synthesis and privacy protection against such prediction*, April 2013.

[20] Y. Yanagisawa, J.-i. Akahani, and T. Satoh, *Shape-based similarity query for trajectory of mobile objects*, in International Conference on Mobile Data Management, Springer, 2003, pp. 63–77.

[21] W. Zaremba and I. Sutskever, *Learning to execute*, CoRR, abs/1410.4615 (2014).

[22] B. D. Ziebart, A. L. Maas, A. K. Dey, and J. A. Bagnell, *Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior*, in

Proceedings of the 10th International Conference on Ubiquitous Computing, UbiComp '08, New York, NY, USA, 2008, ACM, pp. 322–331.

[23] 김은지, 김태욱, 문지형, 조성준, AND 허민회, *딥러닝 기반 개인별 주행 목적지 예측*, 대한산업공학회 춘계공동학술대회 논문집, (2018), pp. 2603–2610.

감사의 글

I first would like to dedicate this thesis to my brother, Yannick Léger, and my father Alain Léger who left us as I was writing this paper. Yannick has always been in his own way an incredibly supportive brother and I owe him a lot. I am also infinitely thankful to my father for everything he has done for me.

I would like to express my sincere gratitude to my advisor Prof. Cho for his support, kindness and sometimes patience during my stay at the Datamining Lab of the Seoul National University.

I obviously thank my mother, Liliane Léger for the unconditional support she provided alongside with my father during all my life, but also to my sister Flora Léger, Inés, and the rest of my family which have always been by my side even in the most difficult moments.

My sincere thanks also goes to my friends Sibylle Benmoufek, Jason Chemin, Fabien Kutle, Valentin Magda, Abhi Ranasinghe, Clémence Rubiella, Lee Jungil, Seo Jangwon and several others for their moral support that been of a significant help in the writing of this thesis.

I finally would like to thank the National Institue of Applied Sciences of Toulouse (INSA de Toulouse) for giving me the opportunity to do this master degree here at the Seoul National University. I must especially thank Prof. Monnier, Besse and Dr. Guillouet from the department of Applied Mathematics as well Ms. Virginie Garry and Prof. Danièle Fournier from the office of international affairs.